Iterated Function Systems and Shape Representation

Paul A. Giles

Grey College, University of Durham.

Submitted as PhD thesis, October 1990.

ITERATED FUNCTION SYSTEMS AND SHAPE REPRESENTATION

Paul A. Giles
School of Engineering and Applied Science
University of Durham
PhD. Thesis (October 1990)

ABSTRACT

We propose the use of iterated function systems as an isomorphic shape representation scheme for use in a machine vision environment. A concise description of the basic theory and salient characteristics of iterated function systems is presented and from this we develop a formal framework within which to embed a representation scheme. Concentrating on the problem of obtaining automatically generated two-dimensional encodings we describe implementations of two solutions. The first is based on a deterministic algorithm and makes simplifying assumptions which limit its range of applicability. The second employs a novel formulation of a genetic algorithm and is intended to function with general data input.

Keywords: Machine Vision, Shape Representation, Iterated Function Systems, Genetic Algorithms.

ACKNOWLEDGEMENTS

I wish to thank my supervisors Roberto Garigliano and Alan Purvis for their help and advice over the last three years and in the preparation of this work.

Thanks also to David and Duncan who helped with the mathematics, to Russell for the pictures, and to Ian for giving me somewhere to live during the final few weeks.

This research was made possible by a CASE studentship award with sponsorship from British Aerospace.

CONTENTS

CHAPTER 1: INTRODUCTION 1

1.1 General Vision 1

1.2 Biological Vision 4

1.3 General System Properties 7

1.4 Geometric Representation 9

1.5 Rule-Based Representation 13

1.6 Pictorial/Iconic Representation 14

1.7 IFS Representation 16

1.8 Thesis Structure 18

CHAPTER 2: ITERATED FUNCTION SYSTEMS 20

2.1 Metric Spaces 20

2.2 The Space $\mathcal{H}(\mathbf{X})$ and its Metric 26

2.3 Mappings on a Metric Space 33

2.4 Mappings on $\mathcal{H}(\mathbf{X})$ 37

2.5 Iterated Function Systems and their Properties 39

2.6 Code Space 44

2.7 Iterated Function Systems as Dynamical Systems 50

2.8 Summary 55

CHAPTER 3: ITERATED FUNCTION SYSTEMS AND SHAPE 57

3.1 2D Shape Representation. 57

3.2 Compactness 64

3.3 Stability 65

3.4 Robustness 68

3.5 Attractor rendering 71

3.6 Summary 78

CHAPTER 4: IMPLEMENTATION OF IFS CODING 80

4.1 Coordinate Systems 82

4.2 Mappings 85

4.3 Encoding Algorithm 89

4.4 Program Performance 100

4.5 Conclusions 110

CHAPTER 5: GENETIC ALGORITHMS 113

5.1 A Formal Framework 115

5.2 Schemata 118

5.3 Reproductive Plans 119

5.4 Genetic Operators 122

5.5 Intrinsic Parallelism 125

5.6 Robustness 129

5.7 Limits on Implementations 130

5.8 An Alternative Plan: D_1 133

5.9 Summary 139

CHAPTER 6: IFS ENCODING BY GENETIC ALGORITHM . 141

6.1 Program Parameters 141

6.2 Program Environment 143

6.3 Solution Representation 144

6.4 Implementation of D_1 146

6.5 Fitness Functions 151

6.6 Parameter Settings 154

6.7 Evaluation vs Population 156

6.8 Program Performance 160

6.9 Conclusion 173

CHAPTER 7: CONCLUSION 176

7.1 Research Directions 177

REFERENCES 180

BIBLIOGRAPHY 185

GLOSSARY 190

APPENDIX A 194

APPENDIX B 206

APPENDIX C 212

1 INTRODUCTION

This thesis addresses the problem of finding a suitable shape representation scheme for a general purpose machine vision system and advocates the use of iterated function systems (IFSs). To explain the motivation for this, we begin in this chapter with a discussion of the fundamentals of vision and give a definition of what we mean by a general purpose system. This leads to an examination of the psychophysical data available on biological vision from which we extract a list of properties that could be expected of an artificial system. We then briefly review some contemporary representation schemes, and the systems in which they have been implemented, making comparisons with the theoretical requirements. In the light of the shortcomings of current representations we emphasise the need for isomorphic, pictorial representations and hence introduce the idea of IFS encoding.

1.1 General Vision

Arbib and Hanson (Arbib and Hanson 1988) suggest that the purpose of biological vision is to provide an animal with the information required for it to successfully interact with its environment, a definition which can be extended to include artificial systems so long as the terms 'interact' and 'environment' are given appropriate meanings. The concept of the task environment of a vision system has been discussed by Nevatia (Nevatia 1978) who gives a qualitative

definition of three distinct environment types based upon the number and complexity of objects in a scene, the degree of constraint in the object orientations, and the amount of a priori knowledge about the scene. The definitions are as follows:

TYPE 1: The number and specific identities of the objects in a scene are known, and their position and orientation are highly constrained.

TYPE 2: The set of all possible objects is small and known but the identity of specific objects in a scene is unknown in advance, with positions and orientations only weakly constrained.

TYPE 3: The number of objects in a scene is large and each is not explicitly known. Objects may adopt any physically permissible position and orientation.

We claim that any general vision system will be required to work in complex type three environments such as those presented by natural scenes in the real world. However, before we can make any further statements as to the requirements of a such a system we must consider what is meant by the term 'successful interaction'. If we take success to be the attainment of some goal or goals then the purpose of vision becomes the interpretation of patterns of light intensities and frequencies in order to form a rational plan of action which, according to Tsotsos (Tsotsos 1984), is the definition of knowledge acquisition. Hence we arrive at a definition of vision as the acquisition of world knowledge from images.

Bullock (Bullock 1978) suggests the definition of a general vision system as one which has abilities as wide as those of the human visual system and which is therefore consistent with the idea of a complex type three environment. However, adopting Bullock's definition implies that a general artificial system should be capable of extracting the same amount of knowledge from a scene as a human

by utilising the same information. This in turn implies that the system must be able to reason based upon the visual information it receives and thus requires an internal representation or model of its environment (Koons and McCormick 1987).

Tsotsos (Tsotsos 1984) borrows terminology from the field of psychology to divide the visual process into two stages. The first, called sensation, is the mere acquaintance with a fact, such as the detection of a certain intensity pattern or some similar intrinsic scene attribute. Perception on the other hand is defined as the association of other knowledge with the information provided by sensation. Tsotsos gives the example of an 'edge' which in the context of sensation is the presence in an image of one of a set of certain light intensities patterns, whilst perceptually it is interpreted as a section of a physical entity (object). We therefore arrive at the conclusion that a general vision system cannot simply label sensation but must perceive the objects contained in a scene by the interpretation of image data using model based reasoning.

It is known that the information contained in an image underconstrains scene interpretation and that in general the problem is intractable (Tsotsos 1987). However, biological visual systems demonstrate that good approximate solutions are attainable, and hence it might be assumed that the best way to proceed in designing a machine vision system would be to model it as closely as possible on its biological counterpart. However, it is not certain that biological systems present the only possible solution to the vision problem, or even that they are the best. Further, because of the disparity between the structure and complexity of the human brain and current serial computers, it is entirely possible that the vision

processing ability of the former cannot be efficiently mimicked by the latter. Despite this, it seems justified to examine the way biological systems operate, specifically the human visual system, with the aim of distilling some basic principles (Weisstein and Maguire 1978, Aviad and Lozinskii 1985, Koons and McCormick 1987).

1.2 Biological Vision

Some properties of human visual information representation have been observed by Kosslyn and Schwartz (Kosslyn and Schwartz 1978). They concentrated attention on what they called 'visual images' or the kind of mental pictures we use when imagining objects or scenes that are not currently available for scrutiny. The suggestion is that such visual images are spatial representations in short term memory that are not simply retrieved but are in some way constructed from more fundamental representations in long term memory using conceptual knowledge. In addition it is suggested that this visual imagery portrays information in an implicit pictorial way and not in an explicit symbolic form. It is further suggested that these visual images can be interpreted by processes similar to those used on real images and thus act as an isomorphic representation, which is to say, "one in which the laws and relationships governing the real world objects are inherent in the data structures and operations of the representation" (Fischler 1978).

The evidence on which Kosslyn and Schwartz based these conclusions was the result of a series of experiments on the extraction of information from visual images. The claim that visual images are not retrieved as a whole but are constructed partwise using conceptual knowledge and more abstract representations is supported by the increased time taken to imagine a scene as the complexity

of the scene increases. If each visual image was stored as a whole then the time required to extract it would be the access time of the representation library, and would not depend on the image content. However, Kosslyn and Schwartz cite evidence that this is not observed, and that subjects take longer to generate a complex scene, implying a construction process. That this construction process proceeds by combining objects or scene parts according to knowledge of permissible configurations is evidenced by the fact that we find it easy to imagine scenes that we have not previously witnessed from just a simple verbal description.

Kosslyn and Schwartz demonstrated the spatial nature of visual images by showing that they are bounded as if contained within some display matrix and that if the images are expanded they will overflow the matrix limits. For example, subjects were asked to imagine objects of various sizes viewed from a distance and to visualise the change in the size of the image as it was brought closer. They were then asked to estimate the distance to the object when it began to overflow, that is, when all parts of the object were no longer clearly visible. The findings were that larger objects seemed to overflow at greater distances, and that the angle subtended by each visual image at overflow was constant - both of which are consistent with the idea that the images are spatial entities that are contained within a finite space. Koons and McCormick (Koons and McCormick 1987) cite similar results but give the additional information that the images are approximately circular in extent, are three dimensional, and occupy a visual angle of around twenty five degrees. Data concerning the time taken to expand, contract or rotate visual images leads to the conclusion that they are transformed gradually and pass through intermediate positions as if a given image is being successively refined. Again Koons and McCormick give the quantitative informa-

tion that visual images are rotated at a rate of between 55 and 60 degrees per second.

Evidence for the isomorphic nature of the representation was obtained by Kosslyn and Schwartz by asking subjects to mentally scan between different pairs of points on a visualised map. The findings were that as the distance between the pairs of points increased so did the scan time. A second test involved the use of schematic drawings of faces consisting of a pair of either dark or light eyes at varying distances above a mouth. Subjects were shown a specific drawing and asked to visualise it, with the focus of attention on the mouth. When asked the question of whether the eyes were light or dark, the response time was found to correlate strongly with the distance of the eyes above the mouth. If the subjects were asked to shrink the image before being asked the question then response times decreased, with the opposite effect being observed if the image was first expanded. Kosslyn and Schwartz interpreted this as the subjects scanning upwards from the mouth to the eyes to retrieve the required information, evidence both of the implicit nature of the representation and also, because of the direct relationship between distance and scanning time, isomorphism.

Finally, that visual images can be interpreted by other processes is supported by findings cited by Kosslyn and Schwartz concerning the visualisation of small image parts. It is found that information relating to small parts of a visual image take longer to recover than that for larger parts, regardless of how highly each of these parts is associated with the imagined object, implying a search of the visual image. From their review of the psychophysical data of the human visual system Koons and McCormick conclude that: "The internal representation of imagery uses operations closely related to those operations used in the external or primary perceptual system."

Weisstein and Maguire (Weisstein and Maguire 1978) have analyzed the high level activity of the human visual system by examining the conditions under which visual illusions such as false contours, perceived occlusion, and apparent connectedness occur. They conclude that these phenomena are due to high level processes overriding, filtering or enhancing low-level data and consequently that feedback and top-down control are both present.

1.3 General System Properties

With the working definition that a general vision system must have the same abilities as the human system, and assuming that it is permissible to try and imitate its operation, we can identify the following properties that are required of a shape representation scheme based on the psychophysical data of the previous section:

i. reconstruction of spatial representations form more abstract forms;

ii. models isomorphic with real world objects;

iii. a conceptual partwise description of objects and scenes;

iv. the ability to manipulate representations using processes analogous to those in the real world;

v. the ability to create new models for previously unseen objects.

The need for spatial, isomorphic representations has been discussed by Fischler (Fischler 1978) who emphasises their intrinsic information content. That is, isomorphic representations contain more information than is made explicit. Fischler gives the example of representing knowledge about the distance between

towns in a given region. A common non-isomorphic approach is to tabulate the data as a mileage chart which explicitly gives the distance between each pair of towns, and has the advantage of giving quick access to the required information. However, an isomorphic representation such as a scale map of the region not only contains the mileage information, although in a form which requires access by making measurements on the map, but also data on the distribution of the towns, such as which is farthest north. Such extra information cannot be derived from the non-isomorphic representation and would need to be entered as a separate piece of knowledge. This is the fundamental problem with non-isomorphic representations and as Fischler notes, "one cannot practically make explicit all of the knowledge needed to create a system capable of general purpose vision".

Pentland (Pentland 1986, 1987) has discussed the use of partwise models based upon 'lump of clay' primitives which give rough descriptions of object subparts and their relative orientations whilst avoiding excessive detail, the basis for this approach being the observation that humans use large scale structures as a guide to perception and often overlook small features. Whilst also observing that the partwise description must be isomorphic both metrically and structurally with the object it models, Pentland asserts that the representation must allow the recognition of objects and learn how to describe new objects.

The need for reasoning based on isomorphic models is described by Hayes (Hayes 1985) who suggests that the correct approach to implementing a representation system is that of 'Naïve Physics' whereby the system is given 'common sense' knowledge of general applicability. Further, Hayes argues that the often used approach of assuming restricted environments or 'toy worlds' whilst producing working systems does not yield any information as to the requirements of a general vision system and, as Bullock (Bullock 1978) points out, is thus not

expandable to real world problems. Hayes proposes that a common sense representation of knowledge should have breadth, in that it covers the full range of physically observable properties such as rigidity, colour, reflectivity etc. whilst also being dense in the sense of having a high information content. A final requirement that Hayes imposes on such a representation scheme is that of uniformity in that it is desirable that there is a common formal framework for each type of information. This view is supported by Tsotsos (Tsotsos 1984) who criticises past attempts at machine vision because of the lack of a formalism within which to define, code and manipulate all of the knowledge of the system, with particular reference to the different forms of knowledge representation for high and low level procedures.

Currently, real world object representation schemes fall in to three basic categories namely, geometric, rule based, and pictorial/iconic. We now briefly discuss examples of each representation type, give descriptions of vision systems that have been based on them, and identify the degree to which they correspond with the requirements of the this section.

1.4 Geometric Representation

One of the most common types of geometric representation schemes is that of constructive solid geometry (CSG), in which an object is described as the Boolean combination of a finite set of geometric shape primitives. Implementations usually take the form of a binary tree in which the nodes represent geometric transformations or combining operations (such as union or intersection) to be applied to child nodes, the terminal nodes being associated with volumetric shape primitives (Anderson et al. 1988). Anderson shows that by using a set

of constrained transformations a CSG tree gives a unique description of an object, given a specific choice of primitives. However, it is the choice of primitives that is the major problem with CSG representations. As Fischler (Fischler 1978) notes, it is not possible to describe naturally occurring objects by the use of a small set of simple primitives, and the use of more complex primitives leads to exponentially increasing numbers.

Pentland (Pentland 1986) and Bajcsy and Solina (Bajcsy and Solina 1987) have implemented CSG representation schemes using superquadrics - a family of three dimensional forms, the surfaces of which are defined by the locus swept out by the tip of a three-dimensional vector $\mathcal{X}(\nu,\omega)$, where ν and ω are latitudinal and longitudinal angles respectively. The shape of each superquadric is determined by parameterising the length of $\mathcal{X}$ in terms of two variables α and β. The set of shapes which comprises the family of superquadrics includes the sphere, cube and cylinder, and so as Pentland points out, constitutes a superset of the more typically used primitives. By using in effect only one parameterised primitive the problem of a combinatorial explosion is reduced and the wide range of shapes that are possible allow for reasonably realistic representations. Pentland takes the process one step further by adding texture to the superquadric surfaces by the use of random fractal techniques (Mandelbrot 1982, Pentland 1984).

A well known alternative to CSG representations is that of generalised cylinders developed by Marr and Nishihara (Marr and Nishihara 1978). Marr (Marr 1978) describes the visual process as proceeding through three representational phases starting with the primal sketch which consists of intrinsic image features such as intensity changes and local geometry, then moving through the $2\frac{1}{2}$D sketch which incorporates viewer centered depth information and surface discontinuities, and finishing with a high level description in terms of geometric

object models. It is at this final level that Marr proposes the use of generalised cylinders. The motivation for the representation is the observation that many objects have a natural coordinate axis and so can be well modelled by defining the variation of the objects cross-section along this axis. Hence a generalised cylinder representation consists of a space curve and a cross-sectional template of fixed shape but varying size. The surface of the modelled object is that which is swept out by the boundary of the template as it is drawn along the curve, the specific shape of the object being represented by variations in template size. By describing each articulated subpart of a shape by a generalised cylinder in its own reference frame, a hierarchical part-wise description of objects is obtained. A criticism of this representation scheme is given by Pentland who observes the extreme degree of abstraction present in such models (and hence the motivation for the work with superquadrics), and also by Marr who acknowledges that objects with no obvious axis such as a crumpled newspaper "pose apparently intractable problems".

The generalised cylinder representation is used in the ACRONYM system of Brooks (Brooks 1981). ACRONYM is of interest because of its use of prediction in the recognition process. Working from object descriptions entered by the user, the system generates two graph representations. The first of these is the object graph, the nodes of which are volumetric representations in the form of generalised cylinders, whilst the connecting arcs describe the relationships between parts. The second graph is the restriction graph which has constraints on the volumetric models as nodes, and sub-class inclusion rules as arcs. From these two graphs ACRONYM produces a third, called the prediction graph, which contains information as to the features that an instance of a given object may produce in an image. The nodes of the prediction graph correspond to possible features, and

the connecting arcs to their relationships. Thus the ACRONYM system makes clear the distinction between models of object features and models of the objects producing these features, and uses the latter to produce the former. The construction of the prediction graph requires the use of geometric reasoning techniques and the understanding of the image formation process, and is thus a good example of the use of naïve physics and model-based reasoning as advocated by Hayes.

The suggestive modelling system (SMS) developed by Fisher (Fisher 1987) has a similar representational structure as that used in ACRONYM, but does not use the generalised cylinder representation. Instead the representation scheme is primarily motivated by the need to simplify object recognition and thus describes objects in terms of one-, two-, and three-dimensional primitives chosen for their 'visual saliency'. These consist of space curves which mark shape and reflectance discontinuities, surface patches which correspond to regions of constant principal curvature and which are bounded by space curves, and finally volume elements which are sub-divided into parameterised STICK, PLATE, and BLOB primitives depending on the number of directions of spatial extent. A typical SMS representation of an object consists of a set of characteristic views, each described using combinations of the primary primitive types, and constitutes an explicit model of the information contained in a $2\frac{1}{2}$D sketch constructed from each of the characteristic viewpoints. The limitations of the SMS system arise from the inability to represent smoothly varying shapes other than a cone - other shapes being modelled piecewise - and the difficulty of representing natural objects. However, perhaps the greatest problem facing SMS is the large number of characteristic views needed for typical objects. Fisher reports that many objects require more than fifty characteristic views to be completely represented, but suggests the use

of as few as five of the most significant views to reduce complexity. Even so Fisher acknowledges that SMS models are currently too complex to be implemented efficiently.

To summarise the discussion of geometric representations, they are by their very nature partwise descriptions and isomorphic to the extent that relative proportions, distances, and relations are preserved. Further, implementations such as ACRONYM and SMS demonstrate that they can support the model-based reasoning that is required of a system if it is to emulate human vision. However, even with the use of superquadrics, geometric representations tend to have too much of a 'cartoon' appearance and have difficulty in succinctly representing natural or irregular forms. Finally, geometric representations suffer difficulty in adding new models to a library because of the complexity inherent in deciding which primitives to use in the decomposition.

1.5 Rule-Based Representation

Rule based representation schemes describe objects by listing their defining properties. For example, a rule-based description of an object might consist of a description of the features it produces in an image together with their relative orientations. The prediction graphs produced by the ACRONYM system are rule based descriptions of objects that have been derived from the geometric model. A system that relies wholly on rule based representations is VISIONS developed by Hanson and Riseman (Hanson and Riseman 1978). It employs a hierarchical data structure consisting of 'schemas' which are structures that contain all the information necessary to describe a given entity. The highest level schemas describe whole scenes such as streets, they include information about required

contents like houses and roads, and give constraints on their sizes and locations. The next level of schemas in the hierarchy describe objects, followed by ones for volumes, surfaces, regions, segments, and finally at the lowest level, vertices. The primary advantage of such a representation scheme over a geometrically based one comes in its ability to model entities with poorly defined spatial extent, or of an abstract nature. For example, VISIONS easily copes with concepts such as sky and ground by giving their definitions in terms of permitted orientations with respect to each other and other objects in a scene. It is impossible to conceive of a general vision system that cannot cope with such concepts and so rule based modelling must be at least part of a general representation scheme. However, the rules used by VISIONS contain no more information than that explicitly given and so every piece of knowledge that the system needs must be stated in this way. This is clearly impractical for a system working in anything but the simplest of environments and is demonstrated not to be the way the human vision system works by the psychophysical data. Further as Waugh (Waugh 1989) points out there is a fundamental problem in deciding just what set of rules will concisely and unambiguously describe an object or scene.

1.6 Pictorial/Iconic Representation

Pictorial or iconic models constitute the third and final category of representation schemes in which information is portrayed either diagrammatically or as pictures. The WHISPER system developed by Funt (Funt 1980) uses diagrammatic representations of information with which to reason about the real world for as Funt describes, diagrams present information in a particularly usable form. The diagrams manipulated by WHISPER correspond to temporal snapshots of configurations of objects in a 'block world' and, using a basic knowledge of

physics, the system is able to determine the stability of configurations and hence the way they will develop with time. The diagrams take the physical form of images in a rectangular pixel array. By inspecting the array WHISPER uses its physics knowledge to update the diagram to represent the state of the system after a small time step. Iterative applications of this procedure eventually produce a stable diagram which corresponds to the equilibrium position that would be obtained by objects interacting in the real world. Whilst WHISPER embodies many of the representational and reasoning abilities identified as necessary for a general system, the object models are crude and not stored within the system but given as input.

The Glimpse system of Koons and McCormick (Koons and McCormick 1987) also employs pictorial representations but in the form of directed graphs. Each node of the graph is a visual snapshot or 'glimpse' of an object taken from a certain viewing position. Arcs of the graph specify the change in viewing position necessary to obtain the picture found at connecting nodes. In addition to the purely pictorial information each glimpse also has associated knowledge which Koons and McCormick suggest as a possible basis for a connection to a more symbolic representation. The nature of the pictorial models ensures that the Glimpse representations are truly isomorphic to the real world objects to which they correspond and hence have a high intrinsic information content. However, the need to store large numbers of glimpses limits the practicality of the approach, a problem that is basic to all pictorial representations.

1.7 IFS Representation

The review of vision systems and representation schemes given in the last three sections indicates that most of the concepts derived from the psychophysical data such as isomorphic representations and model based reasoning have been exploited to a greater or lesser extent. However, each basic type of representation appears to have its own shortcomings. Geometric models tend to oversimplify objects, especially ones with complex natural shapes, and experience difficulty in defining a concise set of volumetric primitives. Rule based representations, whilst being a necessary component of a representation scheme are inadequate in themselves due to a lack of isomorphism and intrinsic information content. Pictorial models, whilst being rich in intrinsic information are expensive to store and difficult to manipulate. In an attempt to combine the positive features of both CSG and pictorial models we propose the use of IFSs as a representation scheme for the following reasons:

1. Owing to the work of Barnsley (Barnsley 1985, 1986, 1988, 1989) the theory of IFSs is well understood and provides a firm foundation on which to build a representation scheme. In particular, Barnsley has in theory solved the problem of finding an IFS encoding of a given shape. (It should be noted at this stage that what we are suggesting here is not the imitation of an image for the purposes of data compression, as successfully attempted by Barnsley, but the geometric modelling of individual objects to a resolution approaching picture quality).

2. An IFS representation exhibits similar properties to those observed for the human visual system. For example object models are stored as a simple list of numbers from which an isomorphic representation can be retrieved. (However we make no claims as to the possibility of the IFS representation and that used by

the human visual system being one and the same. Indeed, this is almost certainly not the case since an IFS scheme does not require the gradual displacement of a model in order to achieve rotation as the psychophysical data suggests for the human system).

3. The encoding process decomposes a shape in terms of primitives which are transformations of the shape itself. This results in an essentially recursive definition and avoids the need for a predefined set of primitives and thus avoids some of the problems experienced by CSG schemes in this area.

4. Since the retrieved representations are geometric, they can be combined in the same way as simple shape primitives in a CSG scheme and therefore be used to encode both whole objects and subparts.

5. There is no restriction as to the type of shapes that can be encoded, both natural and artificial objects being described to a level of accuracy limited only by storage space restrictions. The modelled 'shapes' need not even be connected allowing the representation of such things as cloud patterns. Hence IFS encoding is more flexible than either generalised cylinders of superquadrics.

6. The retrieval procedure for a single code allows the rendering of the pictorial representation in any desired orientation and at any scale thus facilitating the manipulation of the models if used as the basis of an 'experimental' reasoning scheme such as that incorporated in the WHISPER system.

7. Owing to the high quality of the representation that is theoretically possible, a single three-dimensional IFS representation of an object could be used to replace the set of images needed for use in pictorial scheme such as that employed by the Glimpse system.

1.8 Thesis Structure

The remainder of this thesis is devoted to an investigation of the practicality of an IFS representation scheme with the primary intent of determining the accessibility of automatically generated encodings. As a simplification we work with only two-dimensional shapes but there is no reason why the techniques developed cannot be extended to three or more dimensions. The thesis structure is as follows:

Chapter two details the mathematics of IFSs starting from basic principles in order to provide a sound theoretical foundation for the rest of the work and to introduce the terminology used. Specifically an IFS is defined and through the theorems of Barnsley its properties are derived. Anyone with an understanding of basic topology need only be concerned with section 2.5 onwards, whilst those familiar with IFS theory and terminology may safely skip the entire chapter, referring back to it as and when necessary.

Chapter three uses IFS theory to introduce a formal framework for the representation of two-dimensional shapes and derives the properties of such a representation scheme based on the mathematics of the previous chapter. Finally, we give a description of how an IFS representation scheme could be used in a vision system in such a way as to incorporate many of the features observed to be present in the human visual system.

Chapter four describes an attempt at an approximate encoding technique based primarily on the modelling of shape boundary information and aimed at determining the ease with which representations can be automatically generated. Included in this chapter is a discussion of the practicalities involved in implementing an IFS representation scheme.

Chapter five outlines the theory of genetic algorithms and gives a discussion of the problems encountered with practical implementations. A modification of the standard algorithm is described which is designed to improve performance and reduce computational complexity.

Chapter six implements the modified genetic algorithm described in chapter five as an attempt at the automatic generation of accurate IFS representations of arbitrary two-dimensional shapes. The performance of the algorithm is investigated and the success of the application is evaluated.

Chapter seven contains the overall conclusions of the thesis which at this stage can be briefly summarised as follows:

1. Iterated function systems do, as suggested, possess the potential for use as the basis of a powerful shape representation scheme.

2. The automatic generation of IFS encodings of general two-dimensional shapes is a practical proposition using a genetic algorithm.

3. A highly efficient and non-arbitrary formulation of a genetic algorithm has been developed.

2 ITERATED FUNCTION SYSTEMS

In the preceding chapter we introduced the idea of using IFSs as a shape representation scheme for a machine vision system. The purpose of this chapter is to present a concise description of the theory and characteristics of an IFS and to introduce the terminology required for later discussions on the properties of the representation and the requirements of encoding implementations.

The mathematical concepts involved are not complex, although it is necessary to make reference to many fundamental theorems of topology in order to give a complete derivation. In the interest of completeness, all the theorems and proofs necessary for an understanding of IFSs and the processes associated with them have been included. Most of the theorems relating directly to IFSs and their properties are due to Barnsley (Barnsley and Demko 1985, Barnsley et al 1986), and the content of this chapter is firmly based upon the first four chapters of Barnsley's excellent book 'Fractals Everywhere' (Barnsley 1988), and in which a more complete treatment of the following can be found.

2.1 Metric Spaces

To begin we recount the basic topological definitions of concepts such as spaces, the points in a space, and the metric distance between points.

Definition 2.1.1 A **space**, denoted by $\mathbf{X}$, is a set. The **points** of a space are the elements of the set.

In future we use the standard notation of $\mathbf{R}^2$ and $\mathbf{R}^3$ to represent two- and three-dimensional Euclidean space respectively. We shall usually denote points in a space by single characters such as x or y, but occasionally use the longer notation of $(x_1, x_2, \ldots, x_n)$, where n is the dimension of a space, when the coordinates of a point need to be made explicit.

Definition 2.1.2 ($\mathbf{X}$,d) denotes a **metric space**. d is a real-valued function $d : \mathbf{X} \times \mathbf{X} \mapsto \mathbf{R}$ which measures the **distance** between a pair of points x,$y \in \mathbf{X}$, and is known as the **metric**. A metric obeys the following axioms:

(i) $d(x, y) = d(y, x) \qquad \forall\ x, y \in \mathbf{X}$;

(ii) $0 < d(x, y) < \infty \qquad \forall\ x, y \in \mathbf{X}$ and $x \neq y$;

(iii) $d(x, x) = 0 \qquad \forall\ x \in \mathbf{X}$;

(iv) $d(x, y) \leq d(x, z) + d(z, y) \qquad \forall\ x, y, z \in \mathbf{X}$.

Axiom (iv) is often referred to as the triangle inequality for obvious reasons.

In general, these axioms permit the definition of more than one metric for a given space, a topic which we return to in chapter four. However, for the rest of this chapter metrics are used primarily to determine whether a pair of points has moved closer together or farther apart as a result of some operation, and the exact form of the function is unimportant. The definitions and theorems of the remainder of this section construct the vocabulary necessary to discuss the properties of spaces and their subsets.

Definition 2.1.3 Let ($\mathbf{X}$,d) be a metric space with $x \in \mathbf{X}$. Let $\epsilon > 0$ be a given real number, then define the set $B(x,\epsilon)$ as:

$$B(x,\epsilon) = \{y \in \mathbf{X} : d(x,y) \leq \epsilon\}.$$

The set $B(x,\epsilon)$ can be thought of as all the points contained within a 'ball' of 'radius' ϵ centered around the point x. Clearly the shape and dimension of the ball will depend on the dimension of the space $\mathbf{X}$ and the form of the metric function d.

Definition 2.1.4 A sequence of points $\{x_n\}_{n=1}^{\infty}$ in a metric space ($\mathbf{X}$,d) is called a **Cauchy sequence** if, for any given real number $\epsilon > 0$, there is an integer $N > 0$ so that:

$$d(x_n, x_m) < \epsilon \qquad \forall\ n,m > N.$$

The above condition is all that is required of a sequence of points in order for it to be classified as a Cauchy sequence. There need not be any deterministic relationship between successive points, nor is there any requirement that each point be unique.

Definition 2.1.5 A sequence of points $\{x_n\}_{n=1}^{\infty}$ in a metric space ($\mathbf{X}$,d) is said to **converge** to a point $x \in \mathbf{X}$ if, for any given real number $\epsilon > 0$, there is an integer $N > 0$ so that:

$$d(x_n, x) < \epsilon \qquad \forall\ n > N.$$

The point $x \in \mathbf{X}$ to which the sequence converges is known as the **limit** of the sequence, and is written as:

$$x = \lim_{n\to\infty} x_n.$$

In terms of definition 2.1.3 this means that all points x_n for $n > N$ are contained within $B(x, \epsilon)$.

Theorem 2.1.1 *If a sequence of points* $\{x_n\}_{n=1}^{\infty}$ *in a metric space* $(\mathbf{X},d)$ *converges to a point* $x \in \mathbf{X}$, *then* $\{x_n\}_{n=1}^{\infty}$ *is a Cauchy sequence.*

Proof – Since the sequence $\{x_n\}_{n=1}^{\infty}$ is convergent, given $\epsilon > 0$ we can find an integer $k > 0$ such that:

$$d(x_n, x) < \frac{\epsilon}{2} \qquad \forall\ n > k.$$

Using the triangle inequality:

$$d(x_n, x_m) \leq d(x_n, x) + d(x, x_m) < \epsilon \qquad \forall\ n, m > k;$$

and so $\{x_n\}_{n=1}^{\infty}$ is a Cauchy sequence.

Definition 2.1.6 A metric space $(\mathbf{X},d)$ is **complete** if every Cauchy sequence $\{x_n\}_{n=1}^{\infty}$ in $\mathbf{X}$ has a limit $x \in \mathbf{X}$.

Definition 2.1.7 Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a metric space $(\mathbf{X},d)$. A point $x \in \mathbf{X}$ is called a **limit point** of $\mathbf{S}$ if there is a sequence of points $\{x_n\}_{n=1}^{\infty}$ and with $x_n \in \mathbf{S} \setminus \{x\}$ such that $\lim_{n-\infty} x_n = x$.

Definition 2.1.8 Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a metric space $(\mathbf{X},d)$. The **closure** of $\mathbf{S}$, denoted by $\bar{\mathbf{S}}$, is defined to be $\bar{\mathbf{S}} = \mathbf{S} \cup \{\text{limit points of } \mathbf{S}\}$. $\mathbf{S}$ is **closed** if it contains all of its limit points, that is to say, $\mathbf{S} = \bar{\mathbf{S}}$.

Definition 2.1.9 Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a metric space $(\mathbf{X},d)$. **S** is **compact** if every infinite sequence $\{x_n\}_{n=1}^{\infty}$ in **S** contains a subsequence having a limit in **S**.

The important concept here is that the infinite sequence need not converge or even be a Cauchy sequence; it could simply be a sequence of points chosen at random from **S**. A subsequence is a set of points $\{x_{N_i}\}_{i=1}^{\infty}$ such that $x_{N_i} = x_n$ for some n, and such that for a pair of points $x_{N_i} = x_n$ and $x_{N_j} = x_m$, then $N_i > N_j$ implies $n > m$. As an analogy consider trying to pack a long thin piece of string into a finite box. The longer the string is made the tighter it must be packed and the more often it must wind back upon itself. Eventually, the string is so tightly packed that it frequently passes close to any chosen point in the box so that starting from one end of the string and following along its length, marks can be made that are successively closer to the chosen point.

Definition 2.1.10 Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a metric space $(\mathbf{X},d)$. **S** is **totally bounded** if, for all real $\epsilon > 0$, there is a finite set of points $\{y_1, y_2, \ldots, y_n\} \subset \mathbf{S}$ such that for all $x \in \mathbf{S}$, $d(x, y_i) < \epsilon$ for some $y_i \in \{y_1, y_2, \ldots, y_n\}$. The set of points $\{y_1, y_2, \ldots, y_n\}$ is called an ϵ-net.

Theorem 2.1.2 *Let $(\mathbf{X},d)$ be a complete metric space and let $\mathbf{S} \subset \mathbf{X}$. Then* **S** *is compact if and only if it is closed and totally bounded.*

Proof – First suppose **S** is closed and totally bounded. Let $\{x_i \in \mathbf{S}\}_{i=1}^{\infty}$ be an any infinite sequence of points in **S**. We construct an ϵ-net, $\{y_1, y_2, \ldots, y_n\} \subset \mathbf{S}$,

with $\epsilon = 1$. It follows from definitions 2.1.3 and 2.1.10 that:

$$\mathbf{S} \subset \bigcup_{j=1}^{n} B(y_j, 1).$$

The ϵ-net contains a finite number of points yet $\{x_i\}_{i=1}^{\infty}$ is infinite and so there must be a point y_k in the ϵ-net for which $B(y_k, 1) = B_1$ contains infinitely many points of the sequence. Choose N_1 so that $x_{N_1} \in B_1$. Clearly $\mathbf{S} \cap B_1$ is totally bounded so we can construct for it an ϵ-net with $\epsilon = 1/2$. Again one point, y_m, of this ϵ-net must contain infinitely many points of the sequence. With $B(y_m, 1/2) = B_2$ choose a point $x_{N_2} \in B_2$ such that $N_2 > N_1$. Continuing in this way, halving the value of ϵ each time, we generate the subsequence $\{x_{N_n}\}_{n=1}^{\infty}$ of the initial sequence $\{x_i\}_{i=1}^{\infty}$. Since,

$$(\mathbf{S} \cap B_1) \supset (\mathbf{S} \cap B_2) \supset (\mathbf{S} \cap B_3) \supset \ldots \supset (\mathbf{S} \cap B_n) \supset \ldots,$$

and given that the radius of the set B_r is 2^{1-r} then:

$$d(x_{N_k}, x_{N_{k+1}}) \leq 2^{2-k} \qquad \forall\ k \geq 1.$$

Given a real number $\delta > 0$,

$$d(x_{N_k}, x_{N_{k+1}}) < \delta \qquad \forall\ k > (2 - ln(\delta)/ln(2)),$$

so $\{x_{N_n}\}_{n=1}^{\infty}$ is a Cauchy sequence which, using the closure of $\mathbf{S}$, has a limit $x \in \mathbf{S}$. Therefore, $\mathbf{S}$ is compact if it is closed and totally bounded.

To complete the proof, suppose $\mathbf{S}$ is compact but that for $\epsilon > 0$ there does not exist an ϵ-net for $\mathbf{S}$. Then there is an infinite sequence of points $\{x_i \in \mathbf{S}\}_{i=1}^{\infty}$ with $d(x_i, x_j) \geq \epsilon$ for all $i \neq j$. However, due to the compactness of $\mathbf{S}$ this sequence must possess a convergent subsequence with limit in $\mathbf{S}$, and so we can find a pair of integers N_i and N_j with $N_i \neq N_j$ for which $d(x_{N_i}, x_{N_j}) < \epsilon$. We therefore have a contradiction and so an ϵ-net does exist and hence $\mathbf{S}$ is closed and totally bounded. We now have the required result that $\mathbf{S}$ is compact if and only if it is closed and totally bounded.

2.2 The Space $\mathcal{H}(\mathbf{X})$ and its Metric

We now introduce the idea of the space $\mathcal{H}(\mathbf{X})$, the points of which are subsets of the space $\mathbf{X}$, because this is the space in which iterated function systems are defined and later we will show that $\mathcal{H}(\mathbf{R}^2)$ is the space which contains the images used by a machine vision system. We begin by giving a definition of $\mathcal{H}(\mathbf{X})$ and its metric. There follows a derivation of some of the properties of the metric function, and finally we give the proof for the compactness of $\mathcal{H}(\mathbf{X})$.

Definition 2.2.1 Let $(\mathbf{X},d)$ be a complete metric space. Then $\mathcal{H}(\mathbf{X})$ denotes the space whose points are the compact subsets of $\mathbf{X}$ other than the empty set.

Definition 2.2.2 Let $(\mathbf{X},d)$ be a complete metric space with $x \in \mathbf{X}$, and let $\mathbf{B} \in \mathcal{H}(\mathbf{X})$. Define:

$$d(x,\mathbf{B}) = \min\{d(x,y) : y \in \mathbf{B}\}.$$

Then $d(x,\mathbf{B})$ is the distance from the point x to the set $\mathbf{B}$.

Definition 2.2.3 Let $(\mathbf{X},d)$ be a complete metric space and let $\mathbf{A},\mathbf{B} \in \mathcal{H}(\mathbf{X})$. Define the distance from the set $\mathbf{A}$ to the set $\mathbf{B}$ as:

$$d(\mathbf{A},\mathbf{B}) = \max\{d(x,\mathbf{B}) : x \in \mathbf{A}\}.$$

Definition 2.2.4 Let $(\mathbf{X},d)$ be a complete metric space. Then the **Hausdorff distance** between points $\mathbf{A}$ and $\mathbf{B}$ in $\mathcal{H}(\mathbf{X})$ is defined by:

$$h(\mathbf{A},\mathbf{B}) = d(\mathbf{A},\mathbf{B}) \vee d(\mathbf{B},\mathbf{A}).$$

Theorem 2.2.1 *The Hausdorff distance is a metric on the space* $\mathcal{H}(\mathbf{X})$.

Proof – With reference to the axioms of definition 2.1.2, and introducing the binary operator '$\vee$' as taking the maximum of two real numbers, let $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{H}(\mathbf{X})$. Clearly from its definition $h(\mathbf{A}, \mathbf{B}) = h(\mathbf{B}, \mathbf{A})$ and so axiom (i) is satisfied. $h(\mathbf{A}, \mathbf{B}) = d(a, b)$ for some $a \in \mathbf{A}$ and $b \in \mathbf{B}$ and, since $\mathbf{A}$ and $\mathbf{B}$ are compact, then $0 \leq h(\mathbf{A}, \mathbf{B}) < \infty$, and so axiom (ii) is satisfied. We have:

$$h(\mathbf{A}, \mathbf{A}) = d(\mathbf{A}, \mathbf{A}) \vee d(\mathbf{A}, \mathbf{A}) = d(\mathbf{A}, \mathbf{A}) = \max\{d(x, \mathbf{A}) : x \in \mathbf{A}\} = 0.$$

which satisfies axiom (iii). Finally, if $\mathbf{A} \neq \mathbf{B}$ then we can assume that there is a point $a \in \mathbf{A}$ such that $a \notin \mathbf{B}$. Then $h(\mathbf{A}, \mathbf{B}) \geq d(a, \mathbf{B}) > 0$. Therefore, for any $a \in \mathbf{A}$ we have:

$$\begin{aligned} d(a, \mathbf{B}) &= \min\{d(a, b) : b \in \mathbf{B}\}; \\ &\leq \min\{d(a, c) + d(c, b) : b \in \mathbf{B}\} \qquad \forall\ c \in \mathbf{C}; \\ &\leq d(a, c) + \min\{d(c, b) : b \in \mathbf{B}\} \qquad \forall\ c \in \mathbf{C}. \end{aligned}$$

This gives:

$$\begin{aligned} d(a, \mathbf{B}) &\leq \min\{d(a, c) : c \in \mathbf{C}\} + \max\{\min\{d(c, b) : b \in \mathbf{B}\} : c \in \mathbf{C}\}; \\ &= d(a, \mathbf{C}) + d(\mathbf{C}, \mathbf{B}); \end{aligned}$$

and so $d(\mathbf{A}, \mathbf{B}) \leq d(\mathbf{A}, \mathbf{C}) + d(\mathbf{C}, \mathbf{B})$. Similarly $d(\mathbf{B}, \mathbf{A}) \leq d(\mathbf{B}, \mathbf{C}) + d(\mathbf{C}, \mathbf{A})$. Therefore:

$$\begin{aligned} h(\mathbf{A}, \mathbf{B}) &= d(\mathbf{A}, \mathbf{B}) \vee d(\mathbf{B}, \mathbf{A}); \\ &\leq d(\mathbf{B}, \mathbf{C}) \vee d(\mathbf{C}, \mathbf{B}) + d(\mathbf{A}, \mathbf{C}) \vee d(\mathbf{C}, \mathbf{A}); \\ &\leq h(\mathbf{B}, \mathbf{C}) + h(\mathbf{A}, \mathbf{C}); \end{aligned}$$

so satisfying the fourth and final metric axiom.

We now give some properties of the Hausdorff metric that will be neeeded for future proofs.

Lemma 2.2.1 *For all* $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ *and* $\mathbf{E}$ *in* $\mathcal{H}(\mathbf{X})$*:*

$$h(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) \leq h(\mathbf{B}, \mathbf{D}) \vee h(\mathbf{C}, \mathbf{E}).$$

Proof – For $\mathbf{A} \in \mathcal{H}(\mathbf{X})$ we have:

$$d(\mathbf{A} \cup \mathbf{B}, \mathbf{C}) = \max\{d(x, \mathbf{C}) : x \in \mathbf{A} \cup \mathbf{B}\} = \max\{d(x, \mathbf{C}) : x \in \mathbf{A}\} \vee \max\{d(x, \mathbf{C}) : x \in \mathbf{B}\};$$

and so,

$$d(\mathbf{A} \cup \mathbf{B}, \mathbf{C}) = d(\mathbf{A}, \mathbf{C}) \vee d(\mathbf{B}, \mathbf{C}).$$

Also we have:

$$d(\mathbf{A}, \mathbf{B} \cup \mathbf{C}) = \max\{\min\{d(x, y) : y \in \mathbf{B} \cup \mathbf{C}\} : x \in \mathbf{A}\};$$

and we get both $d(\mathbf{A}, \mathbf{B} \cup \mathbf{C}) \leq d(\mathbf{A}, \mathbf{B})$, and $d(\mathbf{A}, \mathbf{B} \cup \mathbf{C}) \leq d(\mathbf{A}, \mathbf{C})$. Consider the distance $d(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E})$. Using the above this becomes:

$$d(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) = d(\mathbf{B}, \mathbf{D} \cup \mathbf{E}) \vee d(\mathbf{C}, \mathbf{D} \cup \mathbf{E}) \leq d(\mathbf{B}, \mathbf{D}) \vee d(\mathbf{C}, \mathbf{E}).$$

and finally:

$$\begin{aligned} h(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) &= d(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) \vee d(\mathbf{D} \cup \mathbf{E}, \mathbf{B} \cup \mathbf{C}); \\ &\leq d(\mathbf{B}, \mathbf{D}) \vee d(\mathbf{C}, \mathbf{E}) \vee d(\mathbf{E}, \mathbf{C}) \vee d(\mathbf{D}, \mathbf{B}); \\ &\leq h(\mathbf{B}, \mathbf{D}) \vee h(\mathbf{C}, \mathbf{E}). \end{aligned}$$

as required.

Definition 2.2.5 Let $\mathbf{S} \subset \mathbf{X}$ and let $\epsilon \geq 0$. Then:

$$\mathbf{S} + \epsilon = \{y \in \mathbf{X} : d(x, y) \leq \epsilon\} \text{ for some } x \in \mathbf{S}.$$

Lemma 2.2.2 *Let* $\mathbf{A}, \mathbf{B} \in \mathcal{H}(\mathbf{X})$ *where* ($\mathbf{X}$,d) *is a metric space. Let a real* $\epsilon > 0$ *be given. Then:*

$$h(\mathbf{A}, \mathbf{B}) \leq \epsilon \quad \Leftrightarrow \quad \mathbf{A} \subset \mathbf{B} + \epsilon \text{ and } \mathbf{B} \subset \mathbf{A} + \epsilon.$$

Proof – $h(\mathbf{A},\mathbf{B}) \leq \epsilon$ implies that $d(\mathbf{A},\mathbf{B}) \leq \epsilon$ and $d(\mathbf{B},\mathbf{A}) \leq \epsilon$. Consider $d(\mathbf{A},\mathbf{B})$ and suppose we have:

$$d(\mathbf{A},\mathbf{B}) = \max\{d(a,\mathbf{B}) : a \in \mathbf{A}\} \leq \epsilon.$$

This implies $d(a,\mathbf{B}) \leq \epsilon$ for all $a \in \mathbf{A}$ and so $a \in \mathbf{B} + \epsilon$ for all $a \in \mathbf{A}$ and hence $\mathbf{A} \subset \mathbf{B} + \epsilon$. Alternatively, suppose $\mathbf{A} \subset \mathbf{B} + \epsilon$. Then for any $a \in \mathbf{A}$ there is a $b \in \mathbf{B}$ such that $d(a,b) \leq \epsilon$. Hence $d(a,\mathbf{B}) \leq \epsilon$ and so $d(\mathbf{A},\mathbf{B}) \leq \epsilon$. The same argument can be applied to $d(\mathbf{B},\mathbf{A})$ to obtain the required result.

In order to prove that $(\mathcal{H}(\mathbf{X}),h)$ is a complete metric space it is necessary first to give what Barnsley calls the 'extension lemma' which is concerned with a property of Cauchy sequences in $\mathcal{H}(\mathbf{X})$.

Lemma 2.2.3 *Let (**X**,d) be a metric space. Let $\{\mathbf{A}_n\}_{n=1}^{\infty}$ be a Cauchy sequence of points in $(\mathcal{H}(\mathbf{X}),h)$. Let $\{n_j\}_{j=1}^{\infty}$ be an infinite sequence of integers such that $0 < n_1 < n_2 < n_3 \ldots$. Suppose we have a Cauchy sequence $\{x_{n_j} \in \mathbf{A}_{n_j}\}_{j=1}^{\infty}$ in (**X**,d), then there is a Cauchy sequence $\{x'_n \in \mathbf{A}_n\}_{n=1}^{\infty}$ such that $x'_{n_j} = x_{n_j}$ for all j.*

Proof – The lemma is obtained by giving a construction for the sequence $\{x'_n\}_{n=1}^{\infty}$ and then proving it has the above properties. For each $n \in \{1,2,\ldots,n_1\}$ choose $x'_n \in \{x \in \mathbf{A}_n : d(x,x_{n_1}) = d(x_{n_1},\mathbf{A}_n)\}$. Hence, x'_n is the closest point, or one of the closest points, in $\mathbf{A}_n$ to x_{n_1}. Similarly choose:

$$x'_n = \{x \in \mathbf{A}_n : d(x,x_{n_j}) = d(x_{n_j},\mathbf{A}_n)\} \qquad \forall\ j \in \{2,3,\ldots\} \text{ and each } n \in \{n_j+1,\ldots,n_{j+1}\}.$$

Clearly from the construction, $x'_{n_j} = x_{n_j}$ and $x'_n \in \mathbf{A}_n$. Let a real $\epsilon > 0$ be given, then we can find a number N_1 for which $d(x_{n_k},x_{n_j}) \leq \frac{\epsilon}{3}$ for all $n_k, n_j > N_1$. Similarly,

there is a number N_2 for which $d(\mathbf{A}_m, \mathbf{A}_n) \leq \frac{\epsilon}{3}$ for all $m, n > N_2$. Let $N = \max\{N_1, N_2\}$ so that for $m, n \geq N$ we have:

$$d(x'_m, x'_n) \leq d(x'_m, x_{n_k}) + d(x_{n_k}, x'_n);$$

$$\leq d(x'_m, x_{n_j}) + d(x_{n_j}, x_{n_k}) + d(x_{n_k}, x'_n);$$

where $m \in \{n_{j-1} + .1, \ldots, n_j\}$ and $n \in \{n_{k-1} + 1, \ldots, n_k\}$. Since $h(\mathbf{A}_m, \mathbf{A}_{n_j}) \leq \frac{\epsilon}{3}$ we have $d(x'_m, x_{n_j}) \leq \frac{\epsilon}{3}$ and similarly $d(x_{n_k}, x'_n) \leq \frac{\epsilon}{3}$. This gives $d(x'_m, x'_n) \leq \epsilon$ for all $m, n > N$ and hence the sequence is a Cauchy sequence.

Theorem 2.2.2 *Let (**X**,d) be a complete metric space. Then ($\mathcal{H}$(**X**),h) is a complete metric space. Moreover, if $\{\mathbf{A}_n \in \mathcal{H}(\mathbf{X})\}_{n=1}^{\infty}$ is a Cauchy sequence then $\mathbf{A} = \lim_{n \to \infty} \mathbf{A}_n$ with $\mathbf{A}_n \in \mathcal{H}(\mathbf{X})$ can be characterised as follows:*

$$\mathbf{A} = \{x \in \mathbf{X} : \text{there is a Cauchy sequence } \{x_n \in \mathbf{A}_n\} \text{ that converges to } x\}.$$

Proof – Given that **A** is defined as above, the proof involves showing that convergence occurs, and that **A** is a non-empty compact set. We first show that **A** is non-empty. Take a sequence of integers $N_1 < N_2 < N_3 < \ldots < N_n < \ldots$ such that:

$$h(\mathbf{A}_m, \mathbf{A}_n) < 2^{-i} \qquad \forall\ m, n \geq N_i.$$

We now have $h(\mathbf{A}_{N_1}, \mathbf{A}_{N_2}) < 1/2$ and so from the definition of the Hausdorff metric we can find a pair of points $x_{N_1} \in \mathbf{A}_{N_1}$ and $x_{N_2} \in \mathbf{A}_{N_2}$ for which $d(x_{N_1}, x_{N_2}) < 1/2$. Hence, we can construct an sequence of points $\{x_{N_i} \in \mathbf{A}_{N_i}\}$ such that $d(x_{N_i}, x_{N_{i+1}}) < 2^{-i}$. Then for $m > n \geq k$ we have:

$$d(x_{N_n}, x_{N_m}) \leq d(x_{N_n}, x_{N_{n+1}}) + d(x_{N_{n+1}}, x_{N_{n+2}}) + \ldots + d(x_{N_{m-1}}, x_{N_m});$$

$$< \sum_{i=k}^{\infty} 2^{-i}.$$

By choosing k large enough, we can make the right hand side of the above equation as small as needed, and hence $\{x_{N_i}\}$ is a Cauchy sequence. Using the previous lemma we can construct a Cauchy sequence $\{x'_i \in \mathbf{A}_i\}$ with $x'_{N_i} = x_{N_i}$. Since $\mathbf{X}$ is complete, this sequence has limit in $\mathbf{X}$ which by definition belongs to $\mathbf{A}$. Hence $\mathbf{A}$ is non-empty and consists of the set of all points which are the limits of Cauchy sequences, $\{x_i \in \mathbf{A}_i\}$.

To show that $\mathbf{A}$ is compact it is sufficient to show that it is closed and totally bounded. Suppose $\{a_i \in \mathbf{A}\}$ is a sequence that converges to a point a. For $\mathbf{A}$ to be closed we need to show that $a \in \mathbf{A}$. For each a_i there exists a sequence $\{x_{i,n} \in \mathbf{A}_n\}$ such that $\lim_{n\to\infty} x_{i,n} = a_i$. We can now choose the subsequences, $\{a_{N_i}\}$ such that $d(a_{N_i}, a) < i^{-1}$ and $\{x_{N_i,m_i} \in \mathbf{A}_{m_i}\}$ such that $d(x_{N_i,m_i}, a_{N_i}) < i^{-1}$. Hence we arrive at $d(x_{N_i,m_i}, a) \leq 2i^{-1}$ which tends to zero as $i \to \infty$. Therefore, we can construct a convergent sequence of points $\{z_i \in \mathbf{A}_i\}$ with limit a, which by definition means $a \in \mathbf{A}$, and hence that $\mathbf{A}$ is closed.

Let a real $\epsilon > 0$ be given. Then for $m, n \geq N$ we have:

$$h(\mathbf{A}_n, \mathbf{A}_m) \leq \epsilon \quad \Leftrightarrow \quad \mathbf{A}_m \subset \mathbf{A}_n + \epsilon \text{ for } m \geq n.$$

For $a \in \mathbf{A}$ we have a sequence $\{x_i \in \mathbf{A}_i\}$ that converges to a. Assume that N is large enough so that we also have:

$$d(x_m, a) < \epsilon \qquad \forall\ m \geq N.$$

We have $x_m \in \mathbf{A}_n + \epsilon$ since $\mathbf{A}_m \subset \mathbf{A}_n + \epsilon$. $\mathbf{A}_n$ is complete and therefore closed so we also have $a \in \mathbf{A}_n$ which gives $\mathbf{A} \subset \mathbf{A}_n + \epsilon$ for large enough n. Assume that $\mathbf{A}$ is not totally bounded and hence that no finite ϵ-net exists. This means we can find an infinite sequence $\{a_i \in \mathbf{A}\}$ such that $d(a_i, a_j) \geq \epsilon$ for $i \neq j$. However, we can find an n for which $\mathbf{A} \subset \mathbf{A}_n + \epsilon/3$, and for each $a_i \in \mathbf{A}$ there is a point $y_i \in \mathbf{A}_n$ such

that $d(a_i, y_i) \leq \epsilon/3$. $\mathbf{A}_n$ is compact so some subsequence $\{y_{n_i}\}$ of the sequence $\{y_i\}$ converges and so we can choose points in $\{y_{n_i}\}$ as close together as we wish. In particular we choose the points y_{n_i} and y_{n_j} such that $d(y_{n_i}, y_{n_j}) \leq \epsilon/3$ and we get:

$$d(a_{n_i}, a_{n_j}) \leq d(a_{n_i}, y_{n_i}) + d(y_{n_i}, y_{n_j}) + d(y_{n_j}, a_{n_j}) < \epsilon/3 + \epsilon/3 + \epsilon/3.$$

This contradicts our initial assumption that $d(a_i, a_j) \geq \epsilon$ and hence $\mathbf{A}$ is totally bounded. It has already been shown that $\mathbf{A}$ is closed, and so now we have that it is compact.

Finally we show that $\lim_{n\to\infty} \mathbf{A}_n = \mathbf{A}$. Let $\epsilon > 0$ be given. Then we can find a sequence of integers $N_1 < N_2 < \ldots < N_k < \ldots$ such that:

$$h(\mathbf{A}_i, \mathbf{A}_j) \leq \frac{\epsilon}{2^{k+1}} \qquad \forall\ i, j \geq N_k.$$

We choose $n < N_1$ such that $h(\mathbf{A}_n, \mathbf{A}_{N_1}) \leq \frac{\epsilon}{2}$. Taking $y \in \mathbf{A}_n$ there is a point $x_{N_1} \in \mathbf{A}_{N_1}$ such that $d(y, x_{N_1}) \leq \frac{\epsilon}{2}$. Similarly there is a point $x_{N_2} \in \mathbf{A}_{N_2}$ such that $d(x_{N_1}, x_{N_2}) \leq \frac{\epsilon}{2^2}$. We can now construct the sequence $x_{N_1}, x_{N_2}, x_{N_3}, \ldots$ such that $d(x_{N_k}, x_{N_{k+1}}) \leq \frac{\epsilon}{2^{k+1}}$. The triangle inequality gives:

$$\begin{aligned} d(y, x_{N_k}) &\leq d(y, x_{N_1}) + d(x_{N_1}, x_{N_2}) + \ldots + d(x_{N_{k-1}}, x_{N_k}); \\ &\leq \epsilon \sum_{n=1}^{N_k} \frac{1}{2^n} = \epsilon(1 - 2^{(1-N_k)}); \\ &\leq \epsilon. \end{aligned}$$

Clearly the Cauchy sequence $\{x_{N_k}\}$ converges to a point $a \in \mathbf{A}$ for which $d(y, a) \leq \epsilon$. We thus have $\mathbf{A}_n \subset \mathbf{A} + \epsilon$ for a sufficiently large n. Combining this with the previous result that $\mathbf{A} \subset \mathbf{A}_n + \epsilon$ for a large enough n, we have that $h(\mathbf{A}, \mathbf{A}_n) \leq \epsilon$ for large n. Thus as $n \to \infty$, $\mathbf{A}_n$ converges to $\mathbf{A}$ and the proof is complete.

This also completes the discussion of the properties of the space $\mathcal{H}(\mathbf{X})$ and its metric which has shown that $(\mathcal{H}(\mathbf{X}), h)$ can be treated like any normal complete metric space.

2.3 Mappings on a Metric Space

We now introduce the idea of transforms on a metric space. Again we begin with some basic definitions and then go on to give the definition of a special group of transforms, called contraction mappings, and describe some of their properties.

Definition 2.3.1 A function $f : \mathbf{X}_1 \mapsto \mathbf{X}_2$ from a metric space $(\mathbf{X}_1, d_1)$ into a metric space $(\mathbf{X}_2, d_2)$ is **continuous** if, for each real number $\epsilon > 0$ and $x \in \mathbf{X}_1$, there is a real number $\delta > 0$ so that:

$$d_1(x, y) < \delta \quad \Rightarrow \quad d_2(f(x), f(y)) < \epsilon.$$

Definition 2.3.2 Let $(\mathbf{X},d)$ be a metric space. A **transformation** on $\mathbf{X}$ is a function $f : \mathbf{X} \mapsto \mathbf{X}$ which assigns exactly one point $f(x) \in \mathbf{X}$ to each point $x \in \mathbf{X}$. If $\mathbf{S} \subset \mathbf{X}$ then $f(\mathbf{S}) = \{f(x) : x \in \mathbf{S}\}$. f is **one-to-one** if $x, y \in \mathbf{X}$ with $f(x) = f(y)$ implies $x = y$. f is **onto** if $f(\mathbf{X}) = \mathbf{X}$. f is **invertible** if it is one-to-one and onto and it is then possible to define a transformation $f^{-1} : \mathbf{X} \mapsto \mathbf{X}$ called the **inverse** of f, defined by $f^{-1}(y) = x$, where $x \in \mathbf{X}$ is the unique point such that $y = f(x)$.

As the name suggests an IFS is concerned with the iterative application of functions, for which we give the definition below using Barnsley's notation.

Definition 2.3.3 Let $f : \mathbf{X} \mapsto \mathbf{X}$ be a transformation on a metric space. The **forward iterates** of f are transformations $f^{\circ n} : \mathbf{X} \mapsto \mathbf{X}$ defined by:

$$f^{\circ 0}(x) = x, \quad f^{\circ 1}(x) = f(x), \quad f^{\circ(n+1)}(x) = f(f^{\circ n}(x)) \qquad \forall\ n = 0, 1, 2, \ldots$$

If f is invertible then the **backward iterates** of f are transformations $f^{\circ(-m)}(x)$: $\mathbf{X} \mapsto \mathbf{X}$ defined by:

$$f^{\circ(-1)}(x) = f^{-1}(x), \quad f^{\circ(-m)}(x) = (f^{\circ m})^{-1}(x) \qquad \forall\ m = 1, 2, 3, \ldots$$

We introduce at this point the definition of a two-dimensional affine transformation since we adopt their use in the implementation of IFS encoding programs as described in chapters four and six.

Definition 2.3.4 A transformation $w : \mathbf{R}^2 \mapsto \mathbf{R}^2$ of the form,

$$w(x_1, x_2) = (ax_1 + bx_2 + e, cx_1 + dx_2 + f),$$

where a, b, c, d, e and f are real numbers, is called a two-dimensional **affine transformation**. In future the alternate matrix notation will be used. That is:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = Ax + t.$$

The definitions and theorems in the remainder of this section describe the effects of iterative application of transformations to pairs of points in a metric space and introduces the concept of a contractivity factor which is of fundamental importance both to the theory and as it turns out, to the practice of IFSs.

Definition 2.3.5 Let $f : \mathbf{X} \mapsto \mathbf{X}$ be a transformation on a metric space. A point $x_f \in \mathbf{X}$ such that $f(x_f) = x_f$ is called a **fixed point** of the transformation.

Definition 2.3.6 A transformation $f : \mathbf{X} \mapsto \mathbf{X}$ on a metric space ($\mathbf{X}$,d) is called **contractive** or a **contraction mapping** if there is a constant $0 \leq s < 1$ such

that:

$$d(f(x), f(y)) \leq sd(x, y) \qquad \forall\ x, y \in \mathbf{X}.$$

The number s is called a **contractivity factor** of f.

Lemma 2.3.1 *Let $w : \mathbf{X} \mapsto \mathbf{X}$ be a contraction mapping on the metric space (**X**,d). Then w is continuous.*

Proof – With reference to definition 2.3.1, let a real $\epsilon > 0$ be given. Let $s > 0$ be a contractivity factor for w. Then:

$$d(w(x), w(y)) \leq sd(x, y) < \epsilon;$$

whenever $d(x, y) < \delta$ where $\delta = \frac{\epsilon}{s}$.

Theorem 2.3.1 *Let $f : \mathbf{X} \mapsto \mathbf{X}$ be a contraction mapping on a complete metric space (**X**,d). Then f possesses exactly one fixed point $x_f \in \mathbf{X}$ and further, for any point $x \in \mathbf{X}$, the sequence $\{f^{\circ n}(x) : n = 0, 1, 2, \ldots\}$ converges to x_f. That is:*

$$\lim_{n \to \infty} f^{\circ n}(x) = x_f \qquad \forall\ x \in \mathbf{X}.$$

Proof – Let $x \in \mathbf{X}$ and let $0 \leq s < 1$ be a contractivity factor for f. Then:

$$d(f^{\circ n}(x), f^{\circ m}(x)) \leq s^{m \wedge n} d(x, f^{\circ |n-m|}(x)) \qquad \forall\ n, m = 0, 1, 2, \ldots,$$

(The notation $m \wedge n$ denotes the minimum of the two real numbers.) In particular after repeated application of the triangle inequality we obtain:

$$\begin{aligned} d(x, f^{\circ k}(x)) &\leq d(x, f(x)) + d(f(x), f^{\circ 2}(x)) + \ldots + d(f^{\circ (k-1)}(x), f^{\circ k}(x)); \\ &\leq (1 + s + s^2 + \ldots + s^{k-1}) d(x, f(x)); \\ &\leq (1 - s^k)(1 - s)^{-1} d(x, f(x)); \\ &\leq (1 - s)^{-1} d(x, f(x)) \qquad \text{for } k = 0, 1, 2, \ldots \end{aligned}$$

Substituting into the above we obtain:

$$d(f^{\circ n}(x), f^{\circ m}(x)) \leq s^{m \wedge n}(1-s)^{-1} d(x, f(x));$$

from which it immediately follows that $\{f^{\circ n}(x)\}_{n=0}^{\infty}$ is a Cauchy sequence. Since $\mathbf{X}$ is complete, this Cauchy sequence possesses a limit $x_f \in \mathbf{X}$, and we have $\lim_{n\to\infty} f^{\circ n}(x) = x_f$. Since f is contractive it is continuous and hence:

$$f(x_f) = f(\lim_{n\to\infty} f^{\circ n}(x)) = \lim_{n\to\infty} f^{\circ(n+1)}(x) = x_f.$$

and so x_f is a fixed point of f. Finally, we show there can be no more than one fixed point. Suppose the opposite. Let x_f and y_f be two fixed points of f. Then $x_f = f(x_f)$, $y_f = f(y_f)$ and $d(x_f, y_f) = d(f(x_f), f(y_f)) \leq s d(x_f, y_f)$ giving $(1-s)d(x_f, y_f) \leq 0$ which implies $d(x_f, y_f) = 0$ and hence $x_f = y_f$.

Lemma 2.3.2 *Let (**X**,d) be a complete metric space. Let $f : \mathbf{X} \mapsto \mathbf{X}$ be a contraction mapping with contractivity factor $0 \leq s < 1$, and let the fixed point of f be $x_f \in \mathbf{X}$. Then:*

$$d(x, x_f) \leq (1-s)^{-1} d(x, f(x)) \qquad \forall\ x \in \mathbf{X}.$$

Proof – The distance function $d(a,b)$ for fixed $a \in \mathbf{X}$ is continuous in $b \in \mathbf{X}$. Hence:

$$\begin{aligned} d(x, x_f) = d(x, \lim_{n\to\infty} f^{\circ n}(x)) &= \lim_{n\to\infty} d(x, f^{\circ n}(x)); \\ &\leq \lim_{n\to\infty} \sum_{m=1}^{n} d(f^{\circ(m-1)}(x), f^{\circ m}(x)); \\ &\leq \lim_{n\to\infty} (1 + s + \ldots + s^{n-1}) d(x, f(x)); \\ &\leq (1-s)^{-1} d(x, f(x)). \end{aligned}$$

2.4 Mappings on $\mathcal{H}(\mathbf{X})$

We now describe the extensions that are made to the definitions of the previous section so that they can be applied to the space $\mathcal{H}(\mathbf{X})$, and prepare for the definition of an IFS by considering mappings which are themselves the union of sets of contraction mappings.

Lemma 2.4.1 *Let $w : \mathbf{X} \mapsto \mathbf{X}$ be a continuous mapping on the metric space (**X**,d). Then w maps $\mathcal{H}(\mathbf{X})$ into itself.*

Proof – Let **S** be a nonempty compact subset of **X**. Then clearly the set $w(\mathbf{S}) = \{w(x) : x \in \mathbf{S}\}$ is nonempty. Hence it is just needed to show that $w(\mathbf{S})$ is compact. Let $\{y_n = w(x_n)\}$ be an infinite sequence of points in $w(\mathbf{S})$. Then $\{x_n\}$ is an infinite sequence of points in **S**. Since **S** is compact there is a subsequence $\{x_{N_n}\}$ which converges to a point $\hat{x} \in \mathbf{S}$. The continuity of w implies that $\{y_{N_n} = f(x_{N_n})\}$ is a subsequence of $\{y_n\}$ which converges to $\hat{y} = f(\hat{x}) \in w(\mathbf{S})$. Thus $w(\mathbf{S})$ is compact.

Lemma 2.4.2 *Let $w : \mathbf{X} \mapsto \mathbf{X}$ be a contraction mapping on the metric space (**X**,d) with a contractivity factor s. Then $w : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ defined by:*

$$w(\mathbf{B}) = \{w(x) : x \in \mathbf{B}\} \qquad \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{X});$$

is a contraction mapping on $(\mathcal{H}(\mathbf{X}),h)$ with contractivity factor s.

Proof – From lemma 2.3.1 it follows that $w : \mathbf{X} \mapsto \mathbf{X}$ is continuous. Hence by lemma 2.4.1, w maps $\mathcal{H}(\mathbf{X})$ into itself. Now let $\mathbf{B},\mathbf{C} \in \mathcal{H}(\mathbf{X})$. Then:

$$d(w(\mathbf{B}), w(\mathbf{C})) = \max\{\min\{d(w(x), w(y)) : y \in \mathbf{C}\} : x \in \mathbf{B}\};$$

$$\leq \max\{\min\{sd(x, y) : y \in \mathbf{C}\} : x \in \mathbf{B}\} = sd(\mathbf{B}, \mathbf{C}).$$

Similarly, $d(w(\mathbf{C}), w(\mathbf{B})) \leq sd(\mathbf{C}, \mathbf{B})$. Hence:

$$\begin{aligned} h(w(\mathbf{B}), w(\mathbf{C})) &= d(w(\mathbf{B}), w(\mathbf{C})) \vee d(w(\mathbf{C}), w(\mathbf{B})); \\ &\leq s(d(\mathbf{B}, \mathbf{C}) \vee d(\mathbf{C}, \mathbf{B})); \\ &\leq sh(\mathbf{B}, \mathbf{C}). \end{aligned}$$

Lemma 2.4.3 *Let (**X**,d) be a metric space. Let $\{w_n : n = 1, 2, \ldots, N\}$ be a set of contraction mappings on ($\mathcal{H}$(**X**),h). Let the contractivity factor for w_n be denoted by s_n for each n. Define $W : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ by:*

$$\begin{aligned} W(\mathbf{B}) &= w_1(\mathbf{B}) \cup w_2(\mathbf{B}) \cup \ldots \cup w_N(\mathbf{B}); \\ &= \bigcup_{n=1}^{N} w_n(\mathbf{B}) \qquad \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{X}). \end{aligned}$$

Then W is a contraction mapping with contractivity factor given by:

$$s = \max\{s_n : n = 1, 2, \ldots, N\}.$$

Proof – We give a proof by induction. Let $\mathbf{B}, \mathbf{C} \in \mathcal{H}(\mathbf{X})$ and assume the lemma is true for some W with $N = m$ mappings. Then consider the addition of an extra transform w_{m+1} to construct W' given by:

$$W'(\mathbf{B}) = \bigcup_{n=1}^{m+1} w_n(\mathbf{B}) = \bigcup_{n=1}^{m} w_n(\mathbf{B}) \cup w_{m+1}(\mathbf{B}) = W(\mathbf{B}) \cup w_{m+1}(\mathbf{B}).$$

We now have:

$$h(W'(\mathbf{B}), W'(\mathbf{C})) = h(W(\mathbf{B}) \cup w_{m+1}(\mathbf{B}), W(\mathbf{C}) \cup w_{m+1}(\mathbf{C})).$$

Using lemma 2.2.1 this becomes:

$$\begin{aligned} h(W'(\mathbf{B}), W'(\mathbf{C})) &\leq h(W(\mathbf{B}), W(\mathbf{C})) \vee h(w_{m+1}(\mathbf{B}), w_{m+1}(\mathbf{C})); \\ &\leq sh(\mathbf{B}, \mathbf{C}) \vee s_{m+1}h(\mathbf{B}, \mathbf{C}); \\ &\leq \max\{s, s_{m+1}\}h(\mathbf{B}, \mathbf{C}); \\ &\leq \max\{s_1, s_2, \ldots, s_{m+1}\}h(\mathbf{B}, \mathbf{C}); \end{aligned}$$

and so if the lemma is true for m transforms it is also true for $m+1$. Consider now the case for $m=2$. Using the same argument as above we obtain:

$$h(W(\mathbf{B}),W(\mathbf{C})) \leq h(w_1(\mathbf{B}),w_1(\mathbf{C})) \vee h(w_2(\mathbf{B}),w_2(\mathbf{C}));$$

$$\leq s_1 h(\mathbf{B},\mathbf{C}) \vee s_2 h(\mathbf{B},\mathbf{C});$$

$$\leq \max\{s_1,s_2\} h(\mathbf{B},\mathbf{C}).$$

Thus the lemma is true for $m=2$ and hence for all $m \geq 2$.

2.5 Iterated Function Systems and their Properties

We are now in a position to define exactly what is meant by an iterated functon system and to give a description of some of its properties.

Definition 2.5.1 (Barnsley 1988) An **iterated function system** consists of a complete metric space $(\mathbf{X},d)$ together with a finite set of contraction mappings $w_n : \mathbf{X} \mapsto \mathbf{X}$ with respective contractivity factors s_n for $n = 1,2,\ldots,N$. The notation for an iterated function system is:

$$\{\mathbf{X}, w_n : n = 1,2,\ldots,N\};$$

and its contractivity factor is;

$$s = \max\{s_n : n = 1,2,\ldots,N\}.$$

Theorem 2.5.1 *Let $\{\mathbf{X}, w_n : n = 1,2,\ldots,N\}$ be an iterated function system with contractivity factor s. Then the transformation $W : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ defined by:*

$$W(\mathbf{B}) = \bigcup_{n=1}^{N} w_n(\mathbf{B}) \qquad \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{X});$$

is a contraction mapping on the complete metric space $(\mathcal{H}(\mathbf{X}),h)$ *with contractivity factor* s. *That is:*

$$h(W(\mathbf{B}),W(\mathbf{C})) \leq sh(\mathbf{B},\mathbf{C}) \qquad \forall\ \mathbf{B},\mathbf{C} \in \mathcal{H}(\mathbf{X}).$$

The unique fixed point $\mathbf{A} \in \mathcal{H}(\mathbf{X})$ *obeys*

$$\mathbf{A} = W(\mathbf{A}) = \bigcup_{n=1}^{N} w_n(\mathbf{A}),$$

and is given by $\mathbf{A} = \lim_{n\to\infty} W^{\circ n}(\mathbf{B})$ *for any* $\mathbf{B} \in \mathcal{H}(\mathbf{X})$.

Proof – The proof of the above theorem follows directly from those of theorem 2.3.1 and lemma 2.4.3.

Definition 2.5.2 The fixed point $\mathbf{A} \in \mathcal{H}(\mathbf{X})$ as described in theorem 2.5.1 is called the **attractor** of the iterated function system.

The attractor of an IFS is what we are really interested in since it is a subset of a metric space which is uniquely defined by a set of contraction mappings which in turn can be described by a simple list of numbers. The use that we make of attractors is discussed in more detail in the next chapter. The next two lemmas are used to show that small changes made to the mappings of an IFS result in correspondingly small changes in the attractor.

Lemma 2.5.1 *Let* $(\mathbf{P},d_p)$ *and* $(\mathbf{X},d)$ *be metric spaces, the latter being complete. Let* $w : \mathbf{P} \times \mathbf{X} \mapsto \mathbf{X}$ *be a family of contraction mappings on* $\mathbf{X}$ *with contractivity factor* $0 \leq s < 1$. *That is, for each* $p \in \mathbf{P}$ *and* $x \in \mathbf{X}$, $w(p,x)$ *is a contraction mapping*

on $\mathbf{X}$. *For each fixed* $x \in \mathbf{X}$ *let* w *be continuous on* $\mathbf{P}$. *Then the fixed point of* w *depends continuously on* p. *That is* $x_f : \mathbf{P} \mapsto \mathbf{X}$ *is continuous.*

Proof – Let $x_f(p)$ denote the fixed point of w for fixed $p \in \mathbf{P}$. Let $p \in \mathbf{P}$ and a real $\epsilon > 0$ be given. Then for all $q \in \mathbf{P}$,

$$\begin{aligned} d(x_f(p), x_f(q)) &= d(w(p, x_f(p)), w(q, x_f(q))), \\ &\leq d(w(p, x_f(p)), w(q, x_f(p))) + d(w(q, x_f(p)), w(q, x_f(q))), \\ &\leq d(w(p, x_f(p)), w(q, x_f(p))) + sd(x_f(p), x_f(q)), \end{aligned}$$

which implies,

$$d(x_f(p), x_f(q)) \leq (1-s)^{-1} d(w(p, x_f(p)), w(q, x_f(p))).$$

Since w is continuous on $\mathbf{P}$ we can choose q to give $0 < d_p(p,q) < \delta$ and so that:

$$d(w(p,x), w(q,x)) \leq \epsilon d_p(p,q) \qquad \forall\ x \in \mathbf{X};$$

which gives,

$$d(x_f(p), x_f(q)) \leq \epsilon(1-s)^{-1} d_p(p,q);$$

and so x_f is continuous.

Lemma 2.5.2 *Let* $(\mathbf{X},d)$ *be a metric space. Suppose we have mappings* $w_n : \mathbf{P} \times \mathbf{X} \mapsto \mathbf{X}$ *for* $n = 1, 2, \ldots, N$ *depending continuously on a parameter* $p \in \mathbf{P}$, *where* $(\mathbf{P},d_p)$ *is a compact metric space. That is* $w_n(p,x)$ *depends continuously on* p *for fixed* $x \in \mathbf{X}$. *Then the transformation* $W : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ *defined by:*

$$W(p, \mathbf{B}) = \bigcup_{n=1}^{N} w_n(p, \mathbf{B}) \qquad \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{X});$$

*is also continuous in p. That is, W(p,**B**) is continuous in p for each* $\mathbf{B} \in \mathcal{H}(\mathbf{X})$ *in the metric space* $(\mathcal{H}(\mathbf{X}),h)$.

Proof – Consider the case $N = 1$. For $\mathbf{B} \in \mathcal{H}(\mathbf{X})$ with $p,q \in \mathbf{P}$, and given a real $\epsilon > 0$,

$$d(w_1(p,\mathbf{B}), w_1(q,\mathbf{B})) = \max\{\min\{d(w_1(p,x), w_1(q,y)) : y \in \mathbf{B}\} : x \in \mathbf{B}\},$$

$$\leq \max\{\min\{d(w_1(p,x), w_1(p,y)) + d(w_1(p,y), w_1(q,y)) : y \in \mathbf{B}\} : x \in \mathbf{B}\}.$$

$\mathbf{P} \times \mathbf{B}$ is compact and $w_1 : \mathbf{P} \times \mathbf{B} \mapsto \mathbf{X}$ is continuous. Hence there is a real number $\delta > 0$ so that $d(w_1(p,y), w_1(q,y)) < \epsilon \ \forall \ y \in \mathbf{B}$, whenever $d_p(p,q) < \delta$. So assuming $d_p(p,q) < \delta$ we have:

$$d(w_1(p,\mathbf{B}), w_1(q,\mathbf{B})) < \max\{\min\{d(w_1(p,x), w_1(p,y) + \epsilon\} : y \in \mathbf{B}\} : x \in \mathbf{B}\};$$

$$\leq d(w_1(p,\mathbf{B}), w_1(p,\mathbf{B})) + \epsilon = \epsilon.$$

similarly,

$$d(w_1(q,\mathbf{B}), w_1(p,\mathbf{B})) < \epsilon \qquad \forall \ d_p(p,q) < \delta,$$

and so,

$$h(w_1(p,\mathbf{B}), w_1(q,\mathbf{B})) < \epsilon \qquad \forall \ d_p(p,q) < \delta.$$

Hence $W(p,\mathbf{B})$ is continuous for $N = 1$. Using lemma 2.2.1 it can be seen that for $N > 1$:

$$h(W(p,\mathbf{B}), W(q,\mathbf{B})) \leq \max\{h(w_n(p,\mathbf{B}), w_n(q,\mathbf{B}))\} = \epsilon';$$

$$\leq \epsilon' \qquad \forall \ d_p(p,q) < \delta.$$

Theorem 2.5.2 *Let* $(\mathbf{X},d)$ *be a metric space and let* $\{\mathbf{X}, w_n : n = 1,2,\ldots,N\}$ *be an iterated function system of contractivity s. Let all* w_n *depend continuously on a parameter* $p \in \mathbf{P}$, *where* $\mathbf{P}$ *is a compact metric space. Then the attractor* $\mathbf{A} \in \mathcal{H}(\mathbf{X})$ *depends continuously on* $p \in \mathbf{P}$ *with respect to the Hausdorff metric.*

Proof – The proof of this theorem follows directly from those of lemmas 2.5.1 and 2.5.2.

The next theorem is of fundamental importance to the possibility of using IFSs as a shape representation scheme since it describes how it is possible to find an IFS encoding of any shape.

Theorem 2.5.3 (The Collage Theorem – Barnsley 1985) *Let (**X**,d) be a complete metric space. Let* $\mathbf{L} \in \mathcal{H}(\mathbf{X})$ *and a real* $\epsilon \geq 0$ *be given. Choose an iterated function system* $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ *with contractivity factor* $0 \leq s < 1$ *so that:*

$$h(\mathbf{L}, \bigcup_{n=1}^{N} w_n(\mathbf{L})) \leq \epsilon;$$

where h is the Hausdorff metric. Then:

$$h(\mathbf{L}, \mathbf{A}) \leq \epsilon(1-s)^{-1};$$

where **A** *is the attractor of the iterated function system. Equivalently:*

$$h(\mathbf{L}, \mathbf{A}) \leq (1-s)^{-1} h(\mathbf{L}, \bigcup_{n=1}^{N} w_n(\mathbf{L})) \qquad \forall\ \mathbf{L} \in \mathcal{H}(\mathbf{X}).$$

Proof – The proof is the same as that for lemma 2.3.2 with the appropriate substitutions.

The implications of this theorem can be best appreciated by considering the case for a two-dimensional space. Each mapping of a set can then be thought of as being a reduced size cut-out copy which is placed over the original. Many such mappings make a rough collage of the original set - and hence Barnsley's name for the theorem. Arranging the collage so that it covers the original set

with no holes or overlaps ensures that the attractor of an IFS consisting of the mappings in the collage will be an exact copy of the set. Any holes or overlaps will cause the attractor to differ from the original set by an amount determined by the contractivity factor of the collage. Thus given a set the problem of finding an IFS which has that set as its attractor reduces to the problem of finding a suitable collage. More on this subject is given in the following chapters.

2.6 Code Space

In order to understand iterated function systems more fully, and to be able to explain the methods of obtaining pictures of their attractors used in the next chapter, it is now necessary to introduce the idea of code space. This leads to a method of labeling each point on an attractor, and ultimately lets us view an IFS as a dynamical system.

Definition 2.6.1 Let Σ be the **code space** on N symbols where N is a positive integer. The symbols are the integers $\{1, 2, 3, \ldots, N\}$. A point in the space Σ is a semi-infinite string of symbols. In general we write a point $\sigma \in \Sigma$ as:

$$\sigma = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5 \ldots \qquad \text{where } \sigma_i \in \{1, 2, 3, \ldots, N\}.$$

Theorem 2.6.1 *For $x, y \in \Sigma$ the function,*

$$d_c(x, y) = d_c(x_1x_2x_3 \ldots, y_1y_2y_3 \ldots) = \sum_{i=1}^{\infty} \frac{|x_i - y_i|}{(N+1)^i}$$

is a metric on the space $\mathbf{\Sigma}$.

Lemma 2.6.1 *We show that* d_c *obeys the axioms of definition 2.1.2.* $|x_i - y_i| = |y_i - x_i|$ *and so* $d_c(x,y) = d_c(y,x)$. *Since* $x_i, y_i \in \{1,2,\ldots,N\}$ *then the maximum value of* $|x_i - y_i|$ *is* $N-1$. *Hence, the maximum value of* $d_c(x,y)$ *is:*

$$(N-1)\sum_{i=1}^{\infty}\frac{1}{(N+1)^i} = \frac{(N-1)}{N};$$

and so;

$$0 \leq d_c(x,y) < 1 \qquad \forall\ x,y \in \mathbf{\Sigma}.$$

If $x_i = y_i\ \forall\ i$, *then* $d_c(x,y) = 0$ *and so* $d_c(x,x) = 0$. *Finally,*

$$|x_i - y_i| \leq |x_i - z_i| + |z_i - y_i| \qquad \forall\ x_i, y_i, z_i \in \{1,2,\ldots,N\},$$

and hence:

$$d_c(x,y) \leq d_c(x,z) + d_c(z,y) \qquad \forall\ x,y,z \in \mathbf{\Sigma}.$$

Consider the attractor of an IFS and imagine its collage (as defined in the collage theorem) superimposed upon it. For N mappings (and assuming for the moment no overlapping between mappings) the attractor is divided up into N regions and each can be labelled according to the mapping that covers it. For example, mapping w_1 maps all the points of the attractor into region one, and w_2 maps all the points into region two. If we then consider the sequence of mappings used to generate a point of the attractor we produce a list of numbers $n_1 n_2 n_3 \ldots$. This list of numbers is called an address of that point of the attractor and, from the preceding definitions, can be seen to be a point in the code space of N symbols. The rest of this section is dedicated to a description of how this idea is formalised and to the properties of point addresses. First however, the following lemma is needed which allows the redefinition of the space underlying an IFS.

Lemma 2.6.2 *Let (**X**,d) be a complete metric space. Let* $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ *be an IFS and let* $\mathbf{K} \in \mathcal{H}(\mathbf{X})$. *Then there exist a set* $\mathbf{K}' \in \mathcal{H}(\mathbf{X})$ *such that* $\mathbf{K} \subset \mathbf{K}'$ *and* $w_n : \mathbf{K}' \mapsto \mathbf{K}'$ *for* $n = 1, 2, \ldots, N$. *That is,* $\{\mathbf{K}', w_n : n = 1, 2, \ldots, N\}$ *is an IFS for which the underlying space is compact.*

Proof – We have defined $W : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ as:

$$W(\mathbf{B}) = \bigcup_{n=1}^{N} w_n(\mathbf{B}) \qquad \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{X}).$$

Therefore if we construct $\mathbf{K}'$ as:

$$\mathbf{K}' = \mathbf{K} \cup W^{\circ 1}(\mathbf{K}) \cup W^{\circ 2}(\mathbf{K}) \cup \ldots \cup W^{\circ n}(\mathbf{K}) \cup \ldots.$$

It is immediately clear that $\mathbf{K} \subset \mathbf{K}'$ and that $W(\mathbf{K}') \subset \mathbf{K}'$.

Definition 2.6.2 Let $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ be an IFS. The code space $(\mathbf{\Sigma}, d_c)$ associated with the IFS is the code space on N symbols, with the metric d_c as defined earlier.

Lemma 2.6.3 *Let (**X**,d) be a complete metric space. Let* $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ *be an IFS with a contractivity factors s and attractor* $\mathbf{A}$. *Let* $(\mathbf{\Sigma}, d_c)$ *denote the code space associated with the IFS. For each* $\sigma \in \mathbf{\Sigma}$, $n \in \mathbf{N}$, $x \in \mathbf{X}$, *define:*

$$\phi(\sigma, n, x) = w_{\sigma 1} \circ w_{\sigma 2} \circ \ldots \circ w_{\sigma n}(x).$$

Let $\mathbf{K}$ *denote a nonempty compact subset of* $\mathbf{X}$. *Then there is a real constant D such that:*

$$d(\phi(\sigma, m, x_1), \phi(\sigma, n, x_2)) \leq D s^{m \wedge n} \qquad \forall\ \sigma \in \mathbf{\Sigma} \quad m, n \in \mathbf{N} \quad \text{and } x_1, x_2 \in \mathbf{K}.$$

Proof – Construct the set $\mathbf{K}'$ as in the previous lemma. Without loss of generality we can take $m < n$. We then have:

$$\phi(\sigma, n, x_2) = \phi(\sigma, m, \phi(\omega, n-m, x_2))\,;$$

where $\omega = \sigma_{m+1}\sigma_{m+2}\ldots\sigma_n\ldots \in \Sigma$. Let $x_3 = \phi(\omega, n-m, x_2)$, then x_3 belongs to $\mathbf{K}'$. We can then write:

$$\begin{aligned} d(\phi(\sigma,m,x_1),\phi(\sigma,n,x_2)) &= d(\phi(\sigma,m,x_1),\phi(\sigma,m,x_3)); \\ &\leq s d(w_{\sigma_2}\circ\ldots\circ w_{\sigma_m}(x_1), w_{\sigma_2}\circ\ldots\circ w_{\sigma_m}(x_3)); \\ &\leq s^2 d(w_{\sigma_3}\circ\ldots\circ w_{\sigma_m}(x_1), w_{\sigma_3}\circ\ldots\circ w_{\sigma_m}(x_3)); \\ &\leq s^m d(x_1,x_3) \leq s^m D; \end{aligned}$$

where $D = \max\{d(x_1,x_3) : x_1, x_3 \in \mathbf{K}'\}$, and is finite since $\mathbf{K}'$ is compact.

Theorem 2.6.2 *Let $\{\mathbf{X}, w_n : n = 1,2,\ldots,N\}$ be an IFS on the metric space $(\mathbf{X},d)$. Let $\mathbf{A}$ denote the attractor of the IFS and let $(\mathbf{\Sigma}, d_c)$ denote the associated code space. Then for $\sigma \in \mathbf{\Sigma}$, $n \in \mathbf{N}$, and $x \in \mathbf{X}$,*

$$\phi(\sigma) = \lim_{n\to\infty} \phi(\sigma, n, x)\,,$$

exists, is contained in $\mathbf{A}$, and is independent of x. The function $\phi : \mathbf{\Sigma} \mapsto \mathbf{A}$ defined in this way is continuous.

Proof – Let $\mathbf{K} \in \mathcal{H}(\mathbf{X})$ such that $x \in \mathbf{K}$. Construct the enclosing set $\mathbf{K}'$ defined previously. With W defined as before we have:

$$\mathbf{A} = \lim_{n\to\infty} W^{\circ n}(\mathbf{K}).$$

In particular $\{W^{\circ n}(\mathbf{K})\}_{n=1}^{\infty}$ is a Cauchy sequence in $(\mathcal{H}(\mathbf{X}),h)$ and, since it is clear that $\phi(\sigma,n,x) \in W^{\circ n}(\mathbf{K})$, it follows that if $\lim_{n\to\infty}\phi(\sigma,n,x)$ exists it must belong to $\mathbf{A}$. The limit can be seen to exist since it has been shown that:

$$d(\phi(\sigma,m,x),\phi(\sigma,n,x)) \leq s^{m\wedge n} D \qquad \forall\ x \in \mathbf{K};$$

and by letting m and n tend to infinity the right hand side can be made arbitrarily small. To prove ϕ is continuous observe that:

$$d_c(\sigma,\omega) < \sum_{m=n+1}^{\infty} \frac{N}{(N+1)^m} = \frac{1}{(N+1)^n}.$$

That is, by choosing σ and ω to agree through the first n terms we can limit the value of d_c. Suppose $\epsilon > 0$ is given, then we can choose an n such that $d_c(\sigma,\omega) < \epsilon$. It follows that we can then write:

$$d(\phi(\sigma,m,x),\phi(\omega,m,x)) = d(\phi(\sigma,n,x_1),\phi(\omega,n,x_2));$$

for some pair of points $x_1, x_2 \in \mathbf{K}'$. Using the result of the previous lemma and taking the limit $m \to \infty$ we obtain:

$$d(\phi(\sigma),\phi(\omega)) < s^n D;$$

and so ϕ is continuous.

Definition 2.6.3 Let $\{\mathbf{X}, w_n : n = 1,2,\ldots,N\}$ be an IFS with associated code space $\mathbf{\Sigma}$. Let $\phi : \mathbf{\Sigma} \mapsto \mathbf{A}$ be a continuous function from code space onto the attractor of the IFS as defined above. An **address** of a point $a \in \mathbf{A}$ is any member of the set:

$$\phi^{-1}(a) = \{\omega \in \mathbf{\Sigma} : \phi(\omega) = a\}.$$

This set is called the addresses of a. An IFS is **totally disconnected** if each point on its attractor has a unique address. An IFS is **just touching** if it is not totally disconnected yet its attractor contains a nonempty set $\mathbf{B}$ which is open in the metric space $\mathbf{A}$ and such that:

(i) $w_i(\mathbf{B}) \cap w_j(\mathbf{B}) = \{\}$ $\quad \forall\ i,j \in \{1,2,\ldots,N\}$ with $i \neq j$;

(ii) $\bigcup_{i=1}^{N} w_i(\mathbf{B}) \subset \mathbf{B}$.

An IFS that is neither disconnected nor just touching is **overlapping**.

Lemma 2.6.4 *Let* $\{X, w_n : n = 1, 2, \ldots, N\}$ *be an IFS with invertible maps and attractor* $\mathbf{A}$. *The IFS is totally disconnected if and only if:*

$$w_i(\mathbf{A}) \cap w_j(\mathbf{A}) = \{\} \qquad \forall\ i, j \in \{1, 2, \ldots, N\} \text{ with } i \neq j.$$

Proof – If an IFS is totally disconnected then every point of the attractor has a unique address. Consider two addresses that differ in only the kth digit. These two addresses agree completely before and after the kth digit but must correspond to two different points. Hence we must have that:

$$w_i(x) \neq w_j(x) \qquad \forall\ x \in \mathbf{A} \text{ and } i \neq j \in \{1, 2, \ldots, N\};$$

which implies the condition given above. Finally, assume the given condition is true, but that the IFS is not totally disconnected and that some point on the attractor has more than one address. Consider the digit at which the two addresses first differ. This time we require $w_i(x) = w_j(x)$ for some $i \neq j \in \{1, 2, \ldots, N\}$ and some $x \in \mathbf{A}$, which gives a contradiction and hence there cannot be two points with the same address.

Having established what we mean by the address of a point on the attractor, and thereby having obtained a way of classifying an IFS, we now go on to examine some properties of addresses.

Definition 2.6.4 Let $\mathbf{A}$ be the attractor of an IFS $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$. A point $a \in \mathbf{A}$ is called a **periodic point** of the IFS if there is a finite sequence of numbers $\{\sigma(n) \in \{1, 2, \ldots, N\}\}_{n=1}^{p}$ such that:

$$a = w_{\sigma(p)} \circ w_{\sigma(p-1)} \circ w_{\sigma(p-2)} \circ \ldots \circ w_{\sigma(1)}(a).$$

If a is periodic then the smallest integer p such that the above condition is true is called the **period** of a.

Definition 2.6.5 A point in code space whose symbols are periodic is called a **periodic address**. A point in code space whose symbols are periodic after a finite initial sequence is omitted is called **eventually periodic**.

Theorem 2.6.3 *The attractor of an IFS is the closure of its periodic points.*

Proof – The code space associated with an IFS is the closure of the set of periodic codes. The mapping ϕ is a continuous mapping from the code space to the attractor, hence **A** is the closure of the set of periodic points of the IFS.

2.7 Iterated Function Systems as Dynamical Systems

The code space terminology introduced in the last section is now used to make the connection between an IFS and a dynamical system. This is necessary in order to understand the workings of Barnsley's algorithm for quickly calculating the attractor of an IFS. Once again we begin with some basic definitions.

Definition 2.7.1 A **dynamical system** is a transformation $f : \mathbf{X} \mapsto \mathbf{X}$ on a metric space $(\mathbf{X},d)$. It is denoted by $\{\mathbf{X}; f\}$. The **orbit** of a point $x \in \mathbf{X}$ is the sequence $\{f^{\circ n}(x)\}_{n=0}^{\infty}$.

Definition 2.7.2 Let $\{\mathbf{X}; f\}$ be a dynamical system. A **periodic point** of f is a point $x \in \mathbf{X}$ such that $f^{\circ n}(x) = x$ for some positive integer, n, called the period

of x. The smallest such integer is called the **minimum period** of x. The orbit of a periodic point of f is called a **cycle** of f. The minimum period of a cycle is the number of distinct points it contains.

Definition 2.7.3 Let $\{\mathbf{X}; f\}$ be a dynamical system. A point $x \in \mathbf{X}$ is called an **eventually periodic point** of f if $f^{\circ m}(x)$ is periodic for some positive integer m.

Definition 2.7.4 Let $(\mathbf{X},d)$ be a metric space. A sequence $\{x_n\}_{n=1}^{\infty}$ of points in $\mathbf{X}$ is said to be **dense** in $\mathbf{X}$ if for each point $x \in \mathbf{X}$ there is a subsequence $\{x_{N_i}\}_{i=0}^{\infty}$ which converges to x. In particular, an orbit $\{x_n\}_{n=1}^{\infty}$ of a dynamical system $\{\mathbf{X}; f\}$ is said to be dense in $\mathbf{X}$ if the sequence $\{x_n\}_{n=1}^{\infty}$ is dense in $\mathbf{X}$.

Lemma 2.7.1 *Let $\{\mathbf{X}, w_n : n = 1,2,\ldots,N\}$ be an IFS with attractor $\mathbf{A}$. If the IFS is totally disconnected then for each $n \in \{1,2,\ldots,N\}$ the mapping $w_n : \mathbf{A} \mapsto \mathbf{A}$ is one-to-one.*

Proof – Take two points $a_1, a_2 \in \mathbf{A}$. Assume that $w_n(a_1) = w_n(a_2) = a$ for some $n \in \{1,2,\ldots,N\}$. This would mean that the point a has two addresses, which is impossible unless $a_1 = a_2$, since the IFS is totally disconnected. Hence w_n is onto for all n.

This last lemma now permits the following association between an IFS and a dynamical system to be made.

Definition 2.7.5 Let $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ be an IFS with attractor $\mathbf{A}$. The associated **shift transformation** on $\mathbf{A}$ is the transformation $S : \mathbf{A} \mapsto \mathbf{A}$ defined by:

$$S(a) = w_n^{-1}(a) \qquad \forall\ a \in w_n(\mathbf{A}).$$

The dynamical system $\{\mathbf{A}; S\}$ is called the **shift dynamical system** associated with the IFS.

The next two theorems concerning the accuracy of the calculation of orbits and the number of cycles with a given minimal period are the important results that this section has been aiming for.

Theorem 2.7.1 *Let $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ be an IFS with a contractivity factor s where $0 < s < 1$. Let $\mathbf{A}$ denote the attractor of the IFS and suppose that each of the transformations $w_n : \mathbf{A} \mapsto \mathbf{A}$ is invertible. Let $\{\mathbf{A}; S\}$ denote the associated random shift dynamical system. Let $\{x'_n\}_{n=0}^{\infty} \subset \mathbf{A}$ be an approximate orbit of S such that:*

$$d(x'_{n+1}, S(x'_n)) \leq \theta \qquad \forall\ n \in \{0, 1, 2, \ldots\};$$

for some fixed constant θ with $0 \leq \theta \leq Diam(\mathbf{A})$. Then there exists an exact orbit given by $\{x_n = S^{\circ n}(x_0)\}_{n=0}^{\infty}$ for some $x_0 \in \mathbf{A}$ such that:

$$d(x'_{n+1}, x_{n+1}) \leq \frac{s\theta}{(1-s)} \qquad \forall\ n \in \{0, 1, 2, \ldots\}.$$

$(\mathrm{Diam}(\mathbf{A}) = \max\{d(x, y) : x, y \in \mathbf{A}\})$.

Proof – For $n = 1, 2, 3, \ldots$ let $\sigma_n \in \{1, 2, \ldots, N\}$ be chosen so that $w_{\sigma_1}^{-1}, w_{\sigma_2}^{-1}, w_{\sigma_3}^{-1}, \ldots$ is the actual sequence of inverse maps used to compute $S(x'_0), S(x'_1), S(x'_2), \ldots$. Let

$\phi : \Sigma \mapsto \mathbf{A}$ denote the code space map associated with the IFS. Then define:

$$x_0 = \phi(\sigma_1\sigma_2\sigma_3\ldots).$$

Hence the exact orbit of the point x_0 is given by:

$$\{x_n = S^{\circ n}(x_0) = \phi(\sigma_{n+1}\sigma_{n+2}\ldots)\}_{n=0}^{\infty}.$$

For some large positive integer M both x_M and $S(x'_{M-1})$ belong to $\mathbf{A}$ and we can write:

$$d(S(x_{M-1}), S(x'_{M-1})) \leq \text{Diam}(\mathbf{A});$$

which is finite since $\mathbf{A}$ is compact. $S(x_{M-1})$ and $S(x'_{M-1})$ are both found using the same inverse map and so it follows that:

$$d(x_{M-1}, x'_{M-1}) \leq s\text{Diam}(\mathbf{A}).$$

We now have:

$$\begin{aligned} d(S(x_{M-2}), S(x'_{M-2})) &= d(x_{M-1}, S(x'_{M-2}));\\ &\leq d(x_{M-1}, x'_{M-1}) + d(x'_{M-1}, S(x'_{M-2}));\\ &\leq \theta + s\text{Diam}(\mathbf{A}). \end{aligned}$$

Again the same inverse map is used to calculate each point so:

$$d(x_{M-2}, x'_{M-2}) \leq s(\theta + s\text{Diam}(\mathbf{A})).$$

Applying the above argument k times we obtain:

$$d(x_{M-k}, x'_{M-k}) \leq s\theta + s^2\theta + \ldots + s^{k-1}\theta + s^k\,\text{Diam}(\mathbf{A});$$

and so for any $0 < n < M$ we have:

$$d(x_n, x'_n) \leq s\theta + s^2\theta + \ldots + s^{M-n-1}\theta + s^{M-n}\,\text{Diam}(\mathbf{A}).$$

Taking the limit as $M \to \infty$ we get the result:

$$d(x_n, x'_n) \leq s\theta(1 + s + s^2 + \ldots) = \frac{s\theta}{(1-s)} \qquad \forall\ n \in \{1, 2, \ldots\}.$$

Finally, we derive an expression for the number of cycles of minimum period p that lie on the attractor of an IFS.

Lemma 2.7.2 *Let $\{\mathbf{A}; S\}$ be the shift dynamical system associated with a totally disconnected IFS, $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$. Let $\mathcal{N}(p)$ denote the number of distinct cycles of minimal period p, for $p \in \{1, 2, 3, \ldots\}$. Then:*

$$\mathcal{N}(p) = \left[\frac{N^p}{p} - \sum_{\substack{k=1 \\ k \text{ divides } p}}^{p-1} \frac{k\mathcal{N}(k)}{p} \right] \qquad \forall\ p \in \{1, 2, \ldots\}.$$

Proof – Every point on a cycle of minimum period p is the fixed point of some transform Ω given by:

$$\Omega(x) = w_{\sigma_1} w_{\sigma_2} w_{\sigma_3} \ldots w_{\sigma_p}(x) = x.$$

That is, the application of p mappings brings us back to the starting point. For a given p there are N^p possible ways of constructing Ω using N mappings. Since each Ω is a contractive mapping it has a unique fixed point which lies on a cycle of period p. Hence N^p is the number of points that lie on a cycle of period p. However, not all these cycles will be of minimum period p, and those that are not must be of a minimum period that is a factor of p. If k is a factor of p and $C(k)$ denotes the number of points that lie on cycles of minimum period k, then the number of points on cycles of minimum period p is

$$N^p - \sum_{\substack{k=1 \\ k \text{ divides } p}}^{p-1} C(k).$$

Since there are p points on every cycle of period p, then the number of distinct cycles of minimum period p is $1/p$ times the number of points on cycles of minimum period p. Therefore:

$$\mathcal{N}(p) = \left[\frac{N^p}{p} - \sum_{\substack{k=1 \\ k \text{ divides } p}}^{p-1} \frac{C(k)}{p} \right] \qquad \forall\ p \in \{1, 2, \ldots\}.$$

Clearly we have that $C(k) = k\mathcal{N}(k)$ and substituting this into the above gives the required result. Finally, we have $\mathcal{N}(1) = N$ since points of period one must be the fixed points of the individual mappings of the IFS.

This final lemma shows that the number of cycles of period p increases rapidly with p, and so it becomes likely that a point chosen at random on the attractor will be part of a cycle of long period. Further, since the attractor is compact, we can expect the orbit of such a point to be dense in the attractor.

2.8 Summary

In this chapter we have, starting from basic topological principles, related a concise derivation of IFS theory and introduced the terminology required for the discussions in the remainder of this work. Specifically, we have given Barnsley's definition of an IFS as being a set of contraction mappings on a complete metric space with the property that when the union of these mappings is applied iteratively to an arbitrary subset of the space, the resulting sequence of sets converges to a non-empty limit set, called the attractor. Important properties of IFSs have been described such as the continuous dependence of the attractor on the mapping parameters, and Barnsley's collage theorem has been stated which allows the calculation of an IFS for a given subset of the space on which it is defined.

$\mathcal{H}(\mathbf{X})$ has been defined as a space with points corresponding to non-empty compact subsets of an underlying metric space, $(\mathbf{X}, d)$, and proved to be complete if $\mathbf{X}$ is complete. The Hausdorff distance, $h(d)$, has been used as the metric on $\mathcal{H}(\mathbf{X})$ and thus an IFS based on $(\mathcal{H}(\mathbf{X}),h(d))$ has a non-empty compact subset $\mathbf{A} \in \mathcal{H}(\mathbf{X})$ as its attractor.

Finally, we have included the proofs which demonstrate that an IFS can be considered as a dynamical system based upon its inverse mappings, and have described how this leads to an alternate definition of an attractor as the closure of the periodic points of such a system.

The following chapter uses these concepts to formalise the proposed shape representation scheme and to explain its properties.

3 ITERATED FUNCTION SYSTEMS AND SHAPE

Using the theory of the previous chapter we now describe a formal framework within which a two-dimensional shape representation scheme can be constructed. We then discuss, with examples, the theoretical properties of such a representation scheme emphasising those that correspond to features identified as beneficial to a general machine vision system. Finally we give an explanation of the random iteration algorithm (RIA) as developed by Barnsley which allows the rapid rendering of the attractor of an IFS.

3.1 2D Shape Representation.

We begin with a pair of our definitions which make clear what we mean by a two-dimensional shape and how we propose to represent one using an IFS.

Definition 3.1.1 Let $(\mathbf{R}^2, d)$ be a metric space consisting of the Euclidean plane, $\mathbf{R}^2$, and a suitable metric function, d. Let a **shape** be any set $\mathbf{S} \in \mathcal{H}(\mathbf{R}^2)$.

Definition 3.1.2 Let $\{\mathbf{R}^2, w_n : n = 1, 2, \ldots, N\}$ be an iterated function system with the two-dimensional Euclidean plane as the underlying metric space. Then a shape **S** is **represented** by the IFS if:

$$\lim_{n \to \infty} W^{\circ n}(\mathbf{B}) = \mathbf{S} \qquad \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{R}^2);$$

where W is as defined in theorem 2.5.1. That is to say, $\mathbf{S}$ is the attractor of the IFS.

The interpretation of such a representation is obvious. The set of points constituting the attractor are the same set of points occupied by the shape in the plane. There are however, two questions that need answering. Firstly, are shapes as just defined useful for study in a machine vision context, and secondly does there exist an accessible representation for any given shape? To answer these questions we make a couple of reasonable assumptions about the environment and image formation process of a hypothetical vision system. We assume that any real-world objects that we may wish our vision system to work with, such as man-made machine parts or naturally occurring flowers and trees, will be compact subsets of three-dimensional Euclidean space, $\mathbf{R}^3$. Also, we assume that the system will produce a two-dimensional image of a three dimensional scene, and that the image formation process - the mapping from $\mathbf{R}^3$ to $\mathbf{R}^2$ - is continuous. Using these assumptions we state the following lemmas.

Lemma 3.1.1 *Let $\mathbf{O} \in \mathcal{H}(\mathbf{R}^3)$ be a three-dimensional object. Let ψ be a continuous transformation $\psi : \mathbf{R}^3 \mapsto \mathbf{R}^2$. Then:*

$$\psi(\mathbf{O}) = \{\psi(y) : y \in \mathbf{O}\} = \mathbf{S};$$

is the image of $\mathbf{O}$ such that $\mathbf{S} \in \mathcal{H}(\mathbf{R}^2)$.

Proof – Let $\{y_n\}_{n=1}^{\infty}$ be an infinite sequence of points in $\mathbf{O}$. Then $\{x_n = \psi(y_n)\}_{n=1}^{\infty}$ is an infinite sequence of points in $\mathbf{S}$. Since $\mathbf{O}$ is compact, there exists a subsequence $\{y_{N_i}\}_{n=1}^{\infty}$ of the initial sequence that has limit $y \in \mathbf{O}$. Since ψ is continuous

the sequence $\{x_{N_i} = \psi(y_{N_i})\}_{N_i=1}^{\infty}$ has limit $x = \psi(y)$ which is contained in $\mathbf{S}$, and hence $\mathbf{S}$ is compact.

This implies that visual information about the real world gathered by a continuous image formation process takes the form of shapes in the image plane, and hence the study of shape encoding is certainly worthwhile.

Next we show that for any shape, no matter how complex, we can always find a good IFS representation. This is a consequence of the collage theorem (2.5.3) which tells us how to find an IFS given the attractor.

Lemma 3.1.2 *For any shape* $\mathbf{S}$, *and given a real number* $\epsilon > 0$, *there exists an iterated function system* $\{\mathbf{R}^2, w_n : n = 1, 2, \ldots, N\}$ *with attractor* $\mathbf{A}$ *for which,* $h(\mathbf{S}, \mathbf{A}) \leq \epsilon$.

Proof – From the collage theorem we have:

$$h(\mathbf{S}, \mathbf{A}) \leq (1 - s)^{-1} h(\mathbf{S}, W(\mathbf{S})) \qquad \forall\ \mathbf{S} \in \mathcal{H}(\mathbf{R}^2);$$

where W and s have their usual meanings. Since $\mathbf{S}$ is compact it is closed and totally bounded, and so there exists a finite ϵ-net, $\{y_1, y_2, ..., y_N\}$. Since an ϵ-net contains only a finite number of points it is closed and totally bounded and hence we have $\{y_n\} \in \mathcal{H}(\mathbf{R}^2)$. From its definition an ϵ-net has the property:

$$h(\mathbf{S}, \{y_n\}) \leq \epsilon.$$

We now choose mappings such that:

$$w_n(x) = y_n \qquad \forall\ x \in \mathbf{S} \text{ and } n = 1, 2, \ldots, N.$$

Hence $W(\mathbf{S}) = \{y_n\}$ with contractivity factor $s = 0$. Finally then:

$$h(\mathbf{S}, \mathbf{A}) \leq 1^{-1} h(\mathbf{S}, \{y_n\});$$

$$\leq \epsilon.$$

This method of constructing an IFS code is clearly inefficient, since it requires one mapping for every point in the ϵ-net, and so will produce a very large number of mappings for small ϵ. However, the proof only serves to show that a high resolution (small ϵ) code can always be found, and does not claim that this code, or the method employed to find it, should be the basis of a practical implementation.

In general the encoding of a shape will involve the search for a collage comprising of a small number of 'large' mappings, which is to say those that map points over a large area. The process of collage construction is similar to decomposing a shape into a set of primitives, the difference being that we are using only the one fundamental primitive, that of the shape itself. The advantage of this is that it is not required to define and store the description of a set of shape primitives prior to encoding, and no reference need be made to any data source external to the IFS code in order to render a picture of the encoded shape. (This is explained in more detail in the section on attractor rendering). Hence we are able to describe complex shapes using a complex primitive without the problem of the exponential increase in the number of primitive types as described by Fischler (Fischler 1978). Further, we can be sure that our primitive is of a suitable type since a shape has just the right morphology with which to describe itself. For example, an angular shape will have an angular primitive, a smoothly rounded shape will have a rounded primitive, and a fractal shape will have a fractal primitive.

Still it is possible that for very complex shapes it will not be possible to obtain a high resolution encoding without recourse to multiple 'small' mappings. In this case it may be necessary to subdivide the shape into smaller and less complex subshapes, and then to encode each one separately, similar to the representation by parts scheme suggested by Pentland (Pentland 1987). The inherent problem is that of finding an algorithm that will produce high resolution collages without human intervention. Ideally, given a fixed number of mappings, the algorithm should find a close to optimal arrangement that minimises the Hausdorff distance between the collage and the shape it is intended to represent. The development of such an algorithm is discussed primarily in chapter six.

The foregoing has illustrated the point that although a given IFS has a unique attractor, there are an infinite number of IFSs that share the same shape for their attractors. This is a consequence of the fact that there are an infinite number of perfect ($\epsilon = 0$) collages that can be made of a given shape. For example, if we have a perfect collage we can construct another by replacing one of its mappings with a mapping of the whole collage.

This is demonstrated in figure 3.1 where the collages depicted have exactly the same attractor. The multiplicity of codes for each shape can be used to advantage for several reasons as will be discussed later.

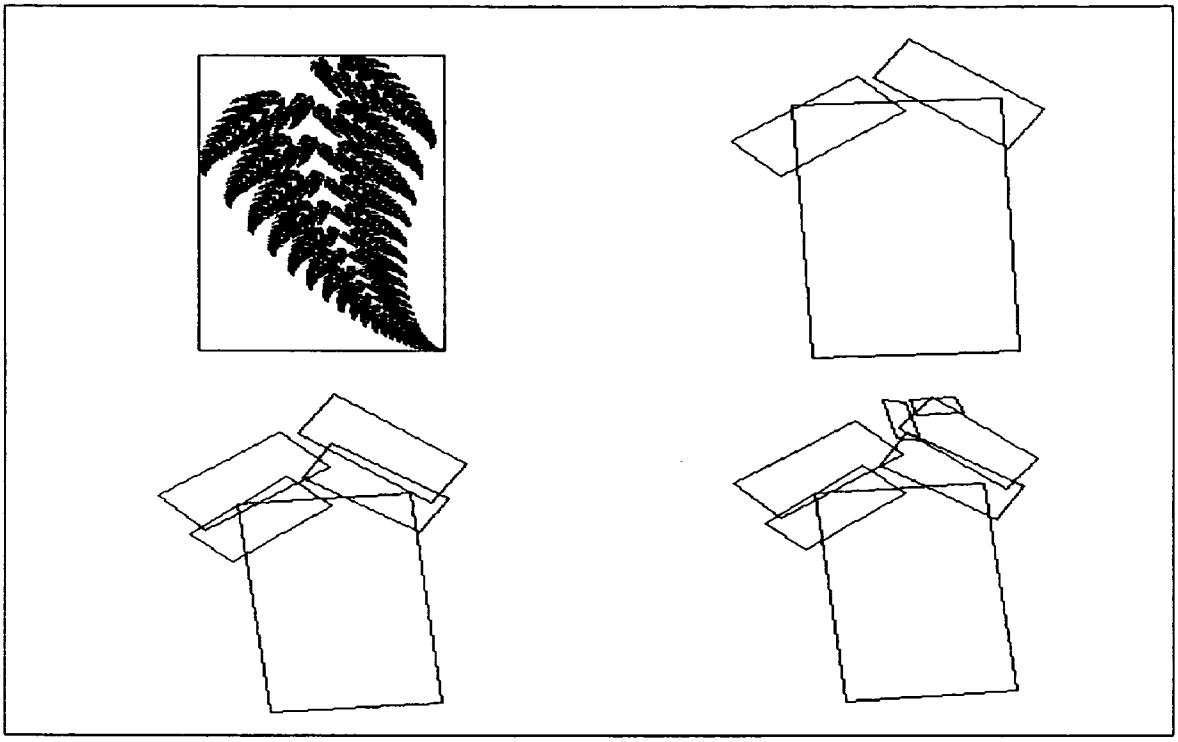


Figure 3.1 *A sequence of collages all having the same fern-like attractor shown in the top left-hand corner. The first collage, top right, is the simplest possible, consisting of just three mappings. The second, bottom left, is comprised of five mappings, the result of replacing the largest mapping of the first collage with its mapping of the whole collage. The final collage, bottom right, has replaced two of the mappings found in the original, and hence contains a total of seven mappings.*

The discussion so far has assumed that the image plane is a continuous space, whereas in reality the image presented to a computer vision system is represented as a two-dimensional pixel array. This does not however, affect our assertion that a good IFS code can be found as our following lemma demonstrates.

Lemma 3.1.3 *Let* $\mathbf{P} \in \mathcal{H}(\mathbf{X})$ *be a finite rectangular array of points in the metric space* $(\mathbf{R}^2, d)$ *such that* $(\mathbf{P},d)$ *is a complete metric space. Let* $\mathbf{S}' = \{p_n \in \mathbf{P} : n = 1,2,\ldots,N\}$ *be a set of points in* $\mathbf{P}$ *such that* $\mathbf{S}'$ *is the discrete approximation of a set* $\mathbf{S} \in \mathcal{H}(\mathbf{R}^2)$. *Then there exist an IFS* $\{\mathbf{P}, w_n : n = 1,2,\ldots,N\}$ *with an attractor* $\mathbf{A}'$ *for which:* $h(\mathbf{A}', \mathbf{S}') = 0$.

Proof – We choose w_n such that:

$$w_n(x) = p_n \qquad \forall\ x \in \mathbf{S}' \text{ and } n = 1,2,\ldots,N.$$

The resulting set of mappings $\{w_n\}$ has contractivity $s = 0$ and $W(\mathbf{S}') = \mathbf{S}'$. Hence, by the collage theorem we have:

$$h(\mathbf{A}', \mathbf{S}') = 1^{-1} h(\mathbf{S}', W(\mathbf{S}')) = h(\mathbf{S}', \mathbf{S}') = 0.$$

It would seem that we can represent exactly any shape defined on a pixel array. However, such an exact encoding amounts to nothing more than a pixel map - a list (albeit in terms of mappings) of the position of every pixel that constitutes the shape. In general we still expect to be able to find high resolution codes, using the collage theorem, that significantly improve upon this.

To summarise this section we have shown that, under some reasonable assumptions, the image of objects in a real-world scene consists of compact subsets of the Euclidean plane and hence are representable as the attractors of two-dimensional IFSs. Further, there exists an IFS representation for every conceivable shape formed in the image plane, and the resolution to which the shape may be encoded is limited only by the number of mappings used.

3.2 Compactness

We now look at the physical form that an IFS representation of a shape would take. Basically an IFS is just a set of mappings which in turn are simply a set of numbers. Hence an IFS code will have the physical form of a list of numbers, and as such has the potential for being a compact way of storing information. The extent of the compactness is dependent on two parameters, the number of mappings used in the IFS, and the number of coefficients required to specify each one. The contractivity term in the bounds on resolution derived from the collage theorem suggests that if we restrict our attention to just-touching IFSs then in general an increase in the number of mappings will produce a corresponding increase in the resolution of the representation, and so in a practical implementation the number of mappings used will be determined by the accuracy to which the system must work. However, we are still free to choose any mappings we want, the only restriction imposed by the mathematics is that they be contractive. Obviously it is desirable to keep the form of the mappings as simple as possible without introducing too many constraints on the range of collages that they are capable of producing. Further, there is nothing in theory to prevent the use of several different types of mapping in the same IFS, although this would needlessly add to the complexity of the resulting code and require more complex decoding algorithms.

		a	b	c	d	e	f
w_1	:	0.50	0.00	0.00	0.50	0.00	18.00
w_2	:	0.50	0.00	0.00	0.50	-15.00	-8.00
w_3	:	0.50	0.00	0.00	0.50	15.00	-8.00

Table 3.1 *The IFS code for a Sierpinski triangle.*

An example of an IFS code for $N = 3$ is given in table 3.1. We have used two-dimensional affine transformations as described in definition 2.3.4. and hence the code consists of three sets of six coefficients. In general, if we require n cofficients to describe each mapping, an IFS will consist of nN floating point numbers.

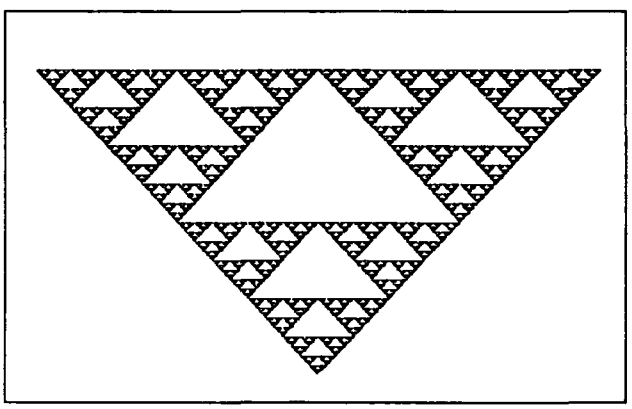


Figure 3.2 *The attractor of the IFS given in table 3.1.*

The attractor of figure 3.2 is a well known fractal called the Sierpinski triangle. Since we already know its IFS representation this and shapes like it will be used to test the collage generating ability of the encoding algorithm described in chapter six.

3.3 Stability

An important property of an IFS coding scheme is stability during the encoding process since, in order for the scheme to be practical, it is necessary that as long as the collage we find is close to the optimum, then the attractor will be similarly close. By defining the term 'close' as meaning close under the Hausdorff metric, we can appeal directly to the collage theorem. It states that if the Hausdorff distance between a shape and its collage is less than or equal to ϵ,

then the distance between the attractor of the IFS associated with the collage and the shape is less than $(1-s)^{-1}\epsilon$, where s is a contractivity factor for the IFS. The stability implied by this can be easily visualised by recalling lemma 2.2.2 which can be interpreted as follows. If the Hausdorff distance between two shapes, **B** and **C**, is ϵ then the shape, created by drawing a circle (for the two-dimensional case) of radius ϵ around every point in **B**, contains the shape **C**, and vice versa. In terms of the collage theorem this means that there is an envelope of size $(1-s)^{-1}\epsilon$ around the coded shape **S** which is guaranteed to contain the attractor. Remembering that the Hausdorff distance between two shapes is simply the distance between some pair of points within the shapes, then it is apparent that not all changes to the collage need change the size of the envelope, indeed most small changes will simply result in some shift of the attractor within the existing envelope. Hence, there exists a large number of collages all within a short distance of the optimum that yield a good attractor. This is demonstrated by the following sequence of pictures in which the attractors are shown for collages of varying quality and contractivity. We have adopted the method of representing a mapping as used by Horn (Horn 1989) in which we show the effect of a mapping on the bounding rectangle of a shape. Although this does not make clear the operation of reflections, it does adequately depict rotations, scalings and translations.

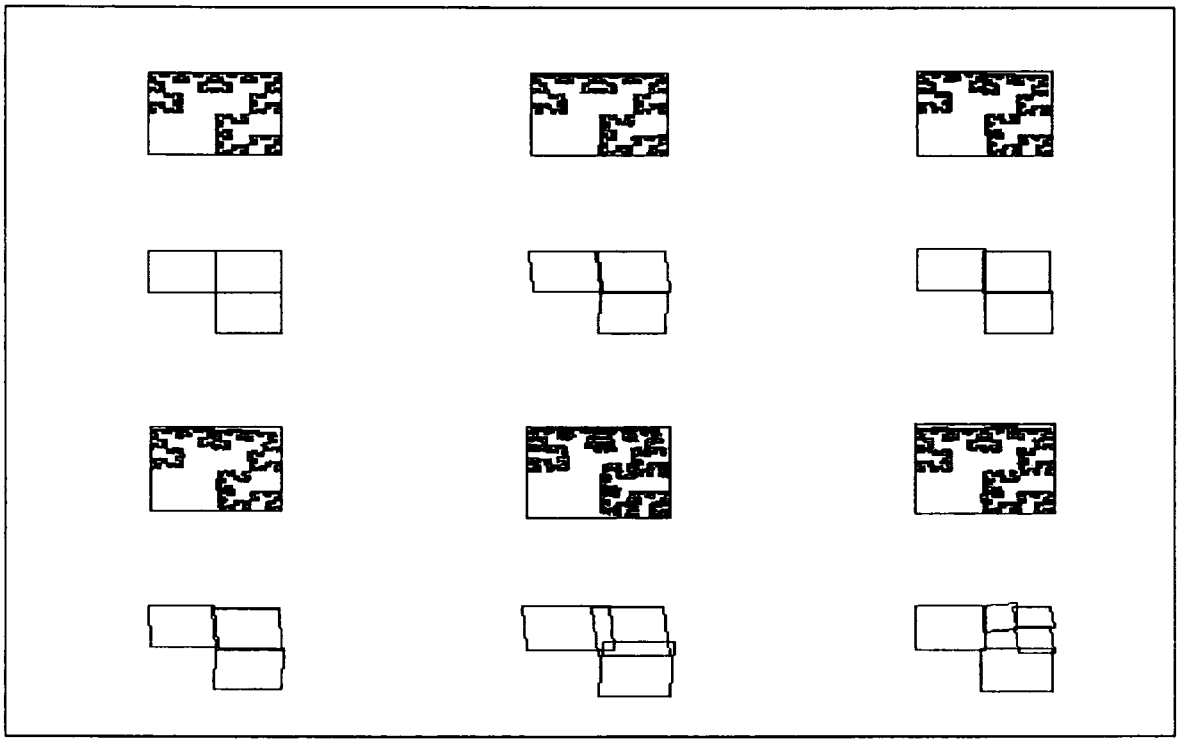


Figure 3.3 *Illustration of how approximate collages give recognizable attractors. (Top left) The attractor obtained with a perfect collage. (Top centre) The attractor for the same collage but with a slight skew introduced into each mapping. (Top right) The effects of slightly translating each mapping. (Bottom left) The attractor for a collage with both skew and translation added to each mapping. (Bottom centre) Skew, translation, and scaling effects. (Bottom right) One of the mappings has been replaced with three smaller mappings which have been skewed, translated and rotated - notice that although the magnitude of the adjustments are the same as before the effects are smaller.*

3.4 Robustness

A further useful property of an IFS code is its robustness with respect to errors introduced into the mapping coefficients. This is predicted by theorem 2.5.2 which states that if the mapping coefficients of an IFS are continuous in some parameter p then the attractor of the IFS is also continuous in p. In practice this means that if a small change is made to one or more of the mapping coefficients, it will produce a correspondingly small change in the attractor. More specifically, if an error in the code means that the maximum value of $h(w_n(\mathbf{S}), w'_n(\mathbf{S}))$ where w'_n denotes an inaccurate mapping, is δ, then the distance between the attractor of the inaccurate code and the exact code is less than or equal to δ. For example, consider the set of IFS codes on the following page and compare their attractors.

		a	b	c	d	e	f
w_1	:	0.60	0.00	0.00	0.60	0.00	-15.00
w_2	:	0.35	0.20	-0.20	0.35	8.00	0.00
w_3	:	0.35	-0.20	0.20	0.35	-8.00	0.00

Table 3.2 *The exact, error free code consisting of three two-dimensional affine transformations.*

		a	b	c	d	e	f
w_1	:	0.60	0.00	0.0**1**	.0.60	0.00	-15.00
w_2	:	0.35	0.20	-0.20	0.35	8.0**3**	0.00
w_3	:	0.35	-0.2**2**	0.20	0.35	-8.00	0.00

Table 3.3 *The same code as in table 3.2 but with small (second decimal place) numerical errors introduced into one coefficient of each mapping. The changed digits are printed in bold type.*

		a	b	c	d	e	f
w_1	:	0.62	0.03	0.12	0.57	-0.20	-15.31
w_2	:	0.32	0.18	-0.21	0.30	7.92	-0.81
w_3	:	0.38	-0.22	0.25	0.36	-8.04	0.29

Table 3.4 *The code of table 3.2 with all of the coefficients affected by small errors.*

		a	b	c	d	e	f
w_1	:	0.65	0.19	0.14	0.04	-0.32	-18.15
w_2	:	0.28	0.21	-0.29	0.21	4.53	-3.16
w_3	:	0.42	-0.29	0.28	0.22	-12.15	2.09

Table 3.5 *The code containing numerous large errors.*

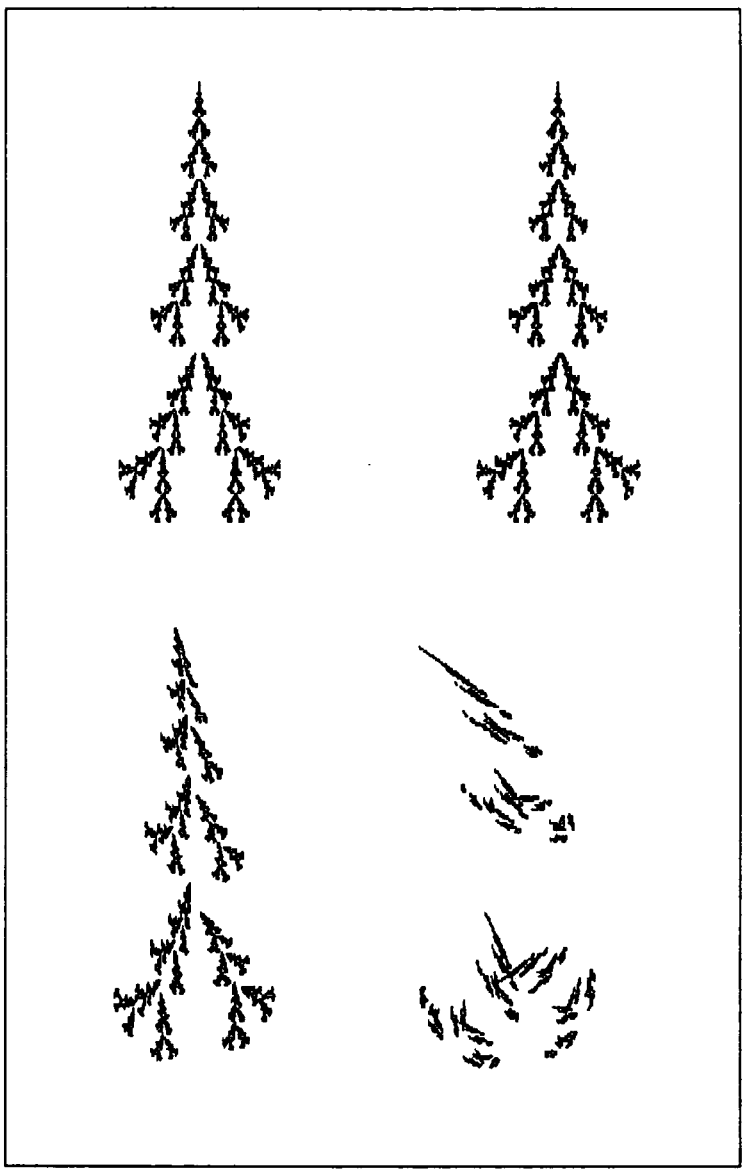


Figure 3.4 *A demonstration of the robustness of the attractor of an IFS with respect to errors introduced into the mapping coefficients. Top left is the attractor for the exact code given in table 3.2. Top right is the attractor for the code in table 3.3 which has several small errors introduced. Bottom left is the attractor for the code in table 3.4 with many small errors, and bottom right is the attractor for the code in table 3.5.*

Clearly a few small errors can be tolerated causing almost no perceptible change in the attractor. Even the introduction of numerous such errors gives results that are, to the human eye at least, recognizable. However, increasing the errors too far results in the break-up of the attractor as a recognizable unit. Since the effect of an error on the attractor is bounded by the size of the effect it has on the mapping in which it appears, less damage is done when errors occur in small mappings. Hence it is possible to build in the desired degree of robustness by limiting the maximum size of the mappings used.

3.5 Attractor rendering

Up to this point we have discussed the use and properties of the attractor of an IFS without describing how to obtain a rendering (picture) of one. This is of crucial importance since it is the attractor and not the IFS itself which displays the stored information in useful form. From the definition of an IFS (definition 2.5.1) we see that the attractor is the limit of the iterative application of the mapping W to an arbitrary starting region. The obvious method of obtaining the attractor then would be to start with some easily defined shape, such as a single point, and iteratively map it under W until the difference between successive iterations became negligible. This illustrates the self-contained nature of an IFS since, due to the recursive way in which it describes shape, there is no need to decode it with reference to any other data.

The method just described has several obvious drawbacks. Firstly, there is no guarantee of the rate of convergence and so it could take a prohibitively large number of iterations before a good approximation to the attractor is obtained. Secondly, even if convergence were to occur quickly, an IFS consisting of a large

number of mappings, or one for which the attractor contained a large number of points, would result in slow rendering. Both these problems would severely restrict the use of an IFS coding scheme in a machine vision system, since we require quick and easy access to stored information.

A much better algorithm is that devised by Barnsley (Barnsley 1988, Barnsley and Sloan 1988). It is based upon the idea of an IFS as a dynamic system and requires the definition of a modified IFS for which a probability is associated with each of the mappings.

Definition 3.5.1 An iterated function system with probabilities consists of an IFS $\{\mathbf{X}, w_n : n = 1, 2, ..., N\}$ together with a set of real numbers $\{p_i : i = 1, 2, ..., N\}$ such that:

$$p_1 + p_2 + p_3 + ... + p_N = 1 \qquad \text{and} \qquad p_i > 0 \qquad \forall\ i.$$

The notation for an iterated function system with probabilities is:

$$\{X, (w_n, p_n) : n = 1, 2, ..., N\}.$$

The rendering method is called the random iteration algorithm (RIA), or as Barnsley sometimes calls it, the chaos game. The algorithm is as follows:

1. Choose an initial point $x_0 \in \mathbf{X}$.
2. Choose one of the mappings of the IFS, w_n, with probability p_n.
3. Apply the selected mapping to the point x_0 and so generate and store the point x_1.
4. At random, select a second mapping, w_m with probability, p_m and apply it to x_1 to generate x_2.
5. Continue in this manner to generate the set of points $\{x_0, x_1, x_2, \ldots, x_K\}$.

6. The set $\mathbf{L} = \{x_n\}_{n=0}^{K}$ for large K is, to a very high probability, a good approximation to the attractor of the IFS.

If the initial point x_0 lies on the attractor then so will all subsequent points. However, if x_0 lies off the attractor it will take several iterations of the algorithm before the points converge onto the attractor, so it is normal practice to discount the first few points of the sequence. To explain why the RIA works, and that almost without exception, it produces a very good approximation of the attractor, we refer back to some IFS theory.

In chapter two we introduced the idea of an IFS as a shift dynamical system on its attractor - that is a dynamical system for which the transformation is defined as $w_n^{-1}(x)$ for all $x \in w_n(\mathbf{A})$. Theorem 2.6.3 demonstrated that the attractor of an IFS is the closure of all the periodic points of the shift dynamical system. In other words, every point on the attractor lies on a periodic orbit, or cycle. Consider a cycle of minimal period p, and let p tend to infinity. The orbit of this cycle consists of an infinite sequence of points on the attractor, and since the attractor is compact, the orbit possess a convergent subsequence. Therefore, we expect cycles of large minimal period to be dense, as defined in definition 2.7.5, and so pass close to every point on the attractor. Hence we can approximate the attractor of an IFS by the orbit of a cycle of large minimal period, K.

Theorem 2.7.3 showed that the number of cycles of minimal period p for the shift dynamical system associated with an IFS of N mappings is given by by the equation:

$$\mathcal{N}(p) = \left[\frac{N^p}{p} - \sum_{\substack{k=1 \\ k \text{ divides } p}}^{p-1} \frac{k\mathcal{N}(k)}{p} \right] \qquad \forall\ p \in \{1, 2, \ldots\}.$$

If we calculate the number of cycles of period $p = 1, 2, \ldots$ for the first few values of N we obtain the data given in the following table.

		Period (p)							
		1	2	3	4	5	6	...	15
(N)	2	2	1	2	3	6	9	...	2182
	3	3	3	8	18	48	116	...	956577
	4	4	6	20	60	204	670	...	71582784
	5	5	10	40	150	624	2580	...	2034505921

Table 3.6 *The number of cycles of minimal period p for the shift dynamical system associated with an IFS of N mappings.*

The figures of table 3.6 show that any cycle of period K, where K is a large number, has a very high probability of being a cycle of minimal period K, and so its orbit will consist of K unique points on the attractor. For example, with $N = 2$ and for p as small as 15 there is a 99.56% chance that any cycle of period 15 we care to construct is a cycle of minimal period 15.

By definition a point on a cycle of period p of a dynamical system $\{\mathbf{X}; f\}$ is a fixed point of $f^{\circ p}$. For a shift dynamical system associated with an IFS, this means that we return to the starting point after the application of a sequence of p inverse mappings. That is:

$$w_{\sigma_1}^{-1} \circ w_{\sigma_2}^{-1} \circ \ldots \circ w_{\sigma_p}^{-1}(x) = x.$$

However, it is clear that we will also arrive back at the starting point after p applications of reverse-order forward mappings. That is:

$$w_{\sigma p} \circ w_{\sigma(p-1)} \circ \ldots \circ w_{\sigma 1}(x) = w_{\sigma p} \circ w_{\sigma(p-1)} \circ \ldots \circ w_{\sigma 1} \circ w_{\sigma 1}^{-1} \circ w_{\sigma 2}^{-1} \circ \ldots \circ w_{\sigma p}^{-1}(x) = x.$$

Hence a sequence of forward mappings is equivalent to the reverse order sequence of inverse mappings and so we can say that a sequence of forward mappings also represents a cycle.

The RIA then works in the following way. It selects a long random sequence of K mappings and takes them as representing a cycle of period K. Since K is large there is a high probability that this random sequence of mappings constitutes a cycle of minimum period K and so its calculated orbit consists of K unique points dense in the attractor. The RIA calculates the points of the orbit according to the algorithm given earlier. To plot the exact orbit denoted by the sequence generated it would appear to be necessary to start with its fixed point. Fortunately this is not true since we are calculating the orbit in terms of forward mappings which are contractive so that any starting point, whether on the attractor or not, will converge very quickly onto the chosen orbit. This is the reason why it may be necessary to discard the first few points generated.

Clearly the RIA is a more efficient algorithm than iterative shape mapping since, (neglecting the first few), every point generated belongs to the attractor, and so the number of points that are rendered is determined by the size of the orbit chosen - in effect the number of iterations of the RIA that we choose to run. With proper implementation it is also possible to obtain rendering rates (pixels per second) that are independent of the number of mappings in the IFS (neglecting program overheads).

A disadvantage of the RIA is that since it is a random process it cannot guarantee to plot any given point. However, we can manipulate the density of points plotted in each region of the attractor by varying the probabilities associated with each mapping. Although a rigorous explanation of this requires a discussion of measures on Borel sets (Barnsley 1985, 1988), it can be seen intuitively that the more often a given mapping is chosen during rendering, the more points that are going to be plotted in the region of the attractor that it covers, and hence the higher the density of points is going to be. However, for

representing shape we are only concerned with whether or not a point belongs to the attractor and so we want an even distribution of points to be produced. To this end we normally select mapping probabilities in proportion to the area of the shape that each mapping covers.

The operation of the random iteration algorithm is shown by the sequence of pictures below. They show the rendering of the attractor of an IFS using increasing values of K. Notice that the basic shape of the attractor is apparent for small values of K and further increases serve only to fill in the fine detail.

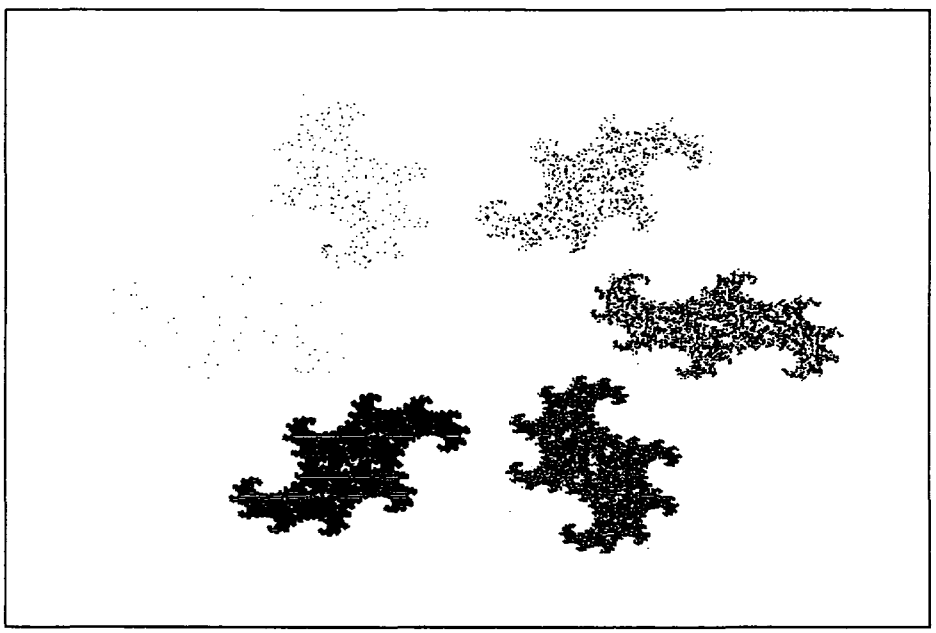


Figure 3.5 *A circle of renderings of the attractor of an IFS consisting of just two mappings. Starting at the nine o'clock position and moving clockwise they correspond to* K *values of* 50, 250, 1000, 4000, 16000, *and* 64000 *respectively.*

The importance of theorem 2.7.1 is now apparent. It states that if there is an error of up to θ in the calculation of each successive point using the random iteration algorithm, then there is always an exact orbit which 'shadows' the

inaccurate one at a distance less than or equal to $s\theta(1-s)^{-1}$. This means that we do not have to be concerned with small rounding or truncation errors when rendering an attractor, since even if we are not plotting the orbit of the cycle that we intended, we are guaranteed to be plotting the orbit of a cycle that is equally good.

A final very important property of the RIA is that it can render a transformation of an attractor almost as easily as it can the basic attractor itself. For example, instead of plotting the set of points $\{x_m\}$ it can instead plot the set $\{\phi(x_n)\}$ where ϕ is the desired transformation. In effect this allows the system to manipulate the representation and obtain any desired view of the encoded information. In two-dimensions this means shapes can be easily rotated or translated, or even scale to see how they would appear from a greater distance. If working in three dimensions, it would allow the construction of any two-dimensional view of a given object. It is this ability to manipulate information in a way isomorphic to that of the real world quantities that has been stressed before as a central requirement of a pictorial representation scheme and it is the relative ease with which this can be achieved using IFS encodings and the RIA that is one of the major advantages of the application. A demonstration of the possibilities of transforming rendered points can be seen by the following picture:

Figure 3.6 *The attractor of the 'fern' IFS as it was originally encoded is shown in the top right-hand corner. The montage of shapes in the centre was constructed entirely from affine transformations of the original attractor, and was produced using the RIA with extra transformations as described in the text.*

3.6 Summary

We have introduced a formal framework in which to embed a two-dimensional IFS shape representation scheme in which shapes are defined as compact subsets of the Euclidean plane produced by images of three-dimensional real world objects. It has been shown that within this scheme every two-dimensional IFS

corresponds to a shape, and further that any shape can be represented by an IFS to an accuracy limited only by the permitted size of the code.

The properties of such a representation scheme such as compactness, stability, robustness, and ease of manipulation have been discussed, and the near picture quality of the encodings demonstrated. Further, we have given an explanation of the operation of the RIA and indicated how this enables the quick rendering of code attractors and enables their easy manipulation.

Regardless of the suitability of an IFS representation scheme based upon its theoretical properties, the question of fundamental importance to the practicality of an implementation is that of the accessibility of automatically generated encodings. This is the problem to which we now turn attention, starting in the next chapter with an attempt at a simplified encoding technique.

4 IMPLEMENTATION OF IFS CODING

We now address what is often described as the 'inverse problem' (Barnsley et al. 1986), which is that of finding an IFS representation for a given shape. Barnsley first proposed a solution based on the moment theory of p-balanced measures (Barnsley and Demko 1985) which relied on a manual approximation of the measures of digitised images, but which was only applicable to a restricted set of shapes (Levy-Vehel and Gagalowicz 1987). With the formulation of the collage theorem, Barnsley developed a technique of general applicability and has employed it in the area of image compression, although the only published work describes an interactive process whereby a tracing of a shape is placed over a computer screen and a software collage construction tool is used by an operator to find the mappings.

Levy-Vehel and Gagalowicz use the collage theorem for shape generation in a computer graphics environment, and employ an optimisation algorithm starting with an essentially random collage and involving the iterative minimisation of some distance function between that collage and the desired shape. Using the Hausdorff distance as the metric they achieved a good result on the single test shape presented, although it was reported that their algorithm failed unless all the mappings of the initial collage intersected the shape. To ensure this condition was satisfied the starting collage was created by hand.

Another attempt at the inverse problem employed the concept of skeletonisation from mathematical morphology (Libeskind-Hadas and Maragos 1987). The

skeleton of a shape is the set of all points which are centres of disks maximal with respect to that shape. (A disk centered at x of radius r is maximal with respect to a shape **S** if it is contained in **S** and is not properly contained in any other disk contained in **S**). Libeskind-Hadas and Maragos describe an interactive system based on the displayed skeleton of a shape which enables the user to identify mappings useful for a collage. The system works well for perfect self-similar fractals, (those which display the same structure at all scales), by enabling the discovery of collages that fit to the shape boundary. However, the system is not intended for use with arbitrary shapes. A simple 'plugging' scheme is proposed to fill shape interiors using circular primitives. The authors suggest that it should be possible to develop a fully automated system, the main difficulty being the detection of useful skeleton branch points.

It appears that the problem of developing a completely automated IFS encoding system is unsolved in that all the above implementations rely on human interaction to a greater or lesser extent. From their results however, we can identify the prime objective of such a system to be the construction of shape collages and, because of the close association between a collage and the IFS it determines, the two terms will often be used interchangeably in the following discussion.

We now describe the implementation of our own algorithm designed for the automatic calculation of shape collages, which is a development of that presented in the paper, (Giles et al. 1989). Following the methodology of Libeskind-Hadas and Maragos, (although not using their skeletonisation technique), the approach taken was to reduce the collage construction process to an essentially one-dimensional problem by looking for mappings which matched to shape boundaries only, thus significantly reducing the search space complexity. The goals of the implementation were thus threefold. Firstly, to determine the practicality of

automatic generation of boundary matching collages assessed upon the criteria of encoding speed, accuracy, and compactness. Secondly, to evaluate the degree to which subsequent plugging of a shape's interior to produce full encodings was possible, and finally to evaluate the use of the implementation as the basis of an IFS shape representation scheme.

To begin, we describe the decisions that need to be made relating to the practical choices presented in selecting coordinate systems, mappings, and an encoding standard.

4.1 Coordinate Systems

In order to calculate mapping coefficients we must first adopt a coordinate system, or frame. We have two basic choices, either to use a space fixed frame such as that provided by the image plane, or a body fixed frame where the coordinates are taken relative to an origin and an axis system defined by the shape itself. This second option is further complicated when we describe shapes in terms of subshapes – should we choose to refer each subshape in a single frame, or should each subshape have its own?

We choose to use body fixed coordinates for the following reason. If the maximum diameter of a given shape is calculated to be D, and the origin of our coordinate system lies somewhere on that shape, then the maximum translational component of a mapping it would ever be necessary to use would be D. This is because we never want a point contained in a shape **S** to be mapped onto a point outside **S**, and since the origin always gets displaced by an amount exactly equal to the translation parameters (the constant terms) of any mapping, this would occur for any mapping of non-zero contractivity and translation of vector length

greater than or equal to D. Thus we put an upper bound on the amount of translation allowed and significantly reduce the search space of possible collages by removing from consideration mappings known to be of no value.

As to the problem of subshapes, Marr (Marr 1978) suggests that the second approach (separate coordinate systems for each articulated subpart) is to be preferred. This has the advantage of allowing the natural description of the relative movements of subparts, and strengthens the isomorphism between the real world object and its representation. However, the current implementation is aimed at encoding only single shapes and so we propose a Cartesian coordinate system with origin at the observed centroid of the shape. The orientation of the axes is not critical, but for simplicity, we take them in the same directions as those of the image plane. The choice of the centroid as origin seems natural since it is easily calculated, relatively stable, and further, corresponds to the centre of mass for uniform planar objects and hence is the point about which they would naturally rotate.

Once the position and orientation of axes are decided, we still have the freedom of choice as to the scale we use. It could be argued that the scale should be taken as $1:1$ with that of the image plane so that rendering of the attractor of the IFS yields the same sized shape as the original. Alternatively, a normalised frame could be chosen in which we set $D = 1$ and hence limit translation parameters to the range $[-1, 1]$ which, as shown later, results in all the mapping parameters being less than unity and giving the code an aesthetic symmetry. This second option is probably more suited to a system working with complex objects and employing subshape coding since a single scale factor stored as part of each code would ensure that they were rendered in the correct proportions to one another. The first option is more suited to an application where shapes

are simple enough to be represented by a single code and it would be beneficial to have their relative sizes reflected in those of their representations. Due to the simplicity of the shapes encoded in this implementation, and the extra computational complexities of using normalised coordinates, we adopt the first option.

The choice of reference frame will not affect the collage or the attractor of the associated IFS, and so it is tempting to think that it will be possible to change the frame of reference, if for example a more natural one was discovered, and simply modify the mapping coefficients so that the IFS is preserved. In general if ϕ is an invertible change of coordinate transformation such that, $\phi(x) = x'$ where primes denote the new coordinate system, then given an IFS $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ in the original coordinate system with attractor $\mathbf{A}$, we have:

$$\bigcup_{n=1}^{N} \phi w_n \phi^{-1}(\mathbf{A}') = \mathbf{A}';$$

gives the collage of the attractor in the new coordinate system. Unfortunately the set of mappings $\{\phi w_n \phi^{-1}\}$ does not constitute an IFS since they are not in general contractive (see figure 4.1 for a graphic example) unless we have the extra constraint:

$$d(\phi(x), \phi(y)) = d(x', y') = kd(x, y) \qquad 0 < k < \infty.$$

That is to say that no independent rescaling of the axes is permitted. Since normalisation of the coordinate system will in general require such rescaling, (as described later), it is not possible to mix easily codes of this format with ones encoded at their natural scale. Should mixing of the two formats be required, all that can be done is to check the mappings produced by the coordinate transformation and ensure that they are all contractive.

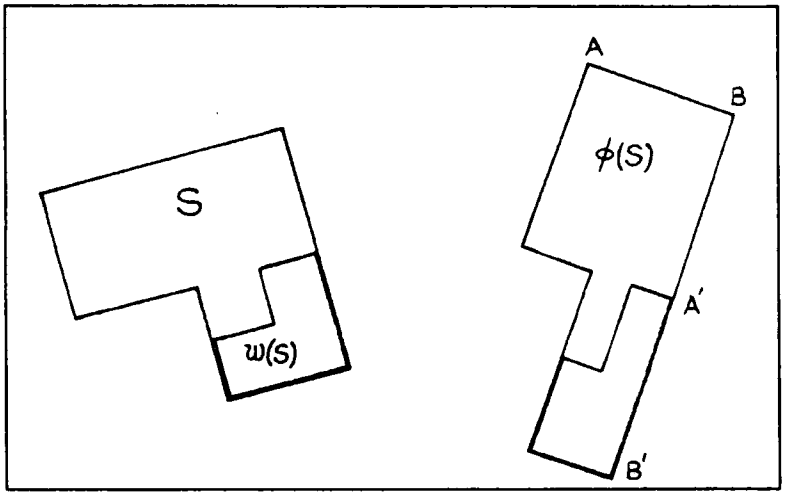


Figure 4.1 *For change of coordinate transformation ϕ the distance $A'B'$ is greater than the distance AB and so the mapping $\phi w \phi^{-1}$ is not contractive.*

We leave a discussion of the metric to be used until the next section since it will be shown to have an effect on the size of mappings we can use.

4.2 Mappings

Once the coordinate system is fixed, we turn attention to the type of mappings to be used. As stated earlier, the only restriction we have is that each must be contractive, and there is no reason why we could not use many different mapping types. However, mixing types within an IFS adds unnecessarily to the complexity of the encoding and rendering algorithms and it is difficult to see what compensating benefits there might be. Obviously we want to keep the form of the mappings as simple as possible since this will both reduce the amount of storage required and improve program speeds, but we must ensure that we have a rich enough palette of maps to produce satisfactory collages.

We propose the use of two-dimensional affine transformations as described in chapter two (definition 2.3.4) and as used by Barnsley. They permit scalings, rotations, reflections, and translations of a shape in the plane, and should therefore provide a broad enough set of collages for our purposes. The general form of an affine transformation requires the use of six coefficients:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}.$$

However, the matrix component of the transformation can be expressed in a different form using the four parameters, r_1, r_2, θ_1 and θ_2 as follows:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} r_1 cos\theta_1 & -r_2 sin\theta_2 \\ r_1 sin\theta_1 & r_2 cos\theta_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}.$$

where x_0 and y_0 are the translation parameters. The value of this alternative notation will become apparent in the following discussion where we examine the limits imposed on parameters by the requirement that a mapping be contractive.

We know from the previous section that the translation parameters can be confined to the range $[-D, D]$ (without axis rescaling) but we can do better than this. If we calculate the extent of the shape in the x- and y-directions relative to the body-fixed reference frame then, because we must not map the origin outside this envelope, we can limit the x and y translations to these values. That is to say, if the shape is bounded by the interval $[x_1, x_2]$ in the x-direction and $[y_1, y_2]$ in the y-direction, then we can restrict x_0 and y_0 such that $x_0 \in [x_1, x_2]$ and $y_0 \in [y_1, y_2]$. For normalised axes we now scale in the x-direction by an amount $\max\{|x_1|, |x_2|\}$ and in the y-direction by $\max\{|y_1|, |y_2|\}$. As for the values of θ_1 and θ_2, we allow them to take values in the range $[0, 2\pi]$. The case for r_1 and r_2 is not quite so simple since all we know is that they scale the shape and so must be bounded by the requirement that the mapping is contractive. Hence we need to derive a contractivity factor for a two-dimensional affine mapping in terms of r_1 and r_2.

We have from definition 2.3.6 that for a contractive mapping, w:

$$d(w(x), w(y)) \leq sd(x, y) \quad \text{where} \quad 0 \leq s < 1.$$

For a two-dimensional affine transformation we have:

$$w(x_1) = ax_1 + bx_2 + e \qquad \text{and} \qquad w(x_2) = cx_1 + dx_2 + f;$$

and using the Euclidean metric we obtain:

$$d(w(x), w(y)) = \sqrt{(a^2 + c^2)(x_1 - y_1)^2 + (b^2 + d^2)(x_2 - y_2)^2 + 2(ab + cd)(x_1 - y_1)(x_2 - y_2)}.$$

However, $(a^2 + c^2) = r_1^2$, $(b^2 + d^2) = r_2^2$ and $(ab + cd) = r_1 r_2 \sin(\theta_1 - \theta_2)$, which gives the condition:

$$d(w(x), w(y)) \leq |r_1||x_1 - y_1| + |r_2||x_2 - y_2|.$$

Now let $r = \max\{|r_1|, |r_2|\}$ so we have:

$$d(w(x), w(y)) \leq r(|x_1 - y_1| + |x_2 - y_2|).$$

Imposing the contractivity constraint gives the condition:

$$\frac{r(|x_1 - y_1| + |x_2 - y_2|)}{\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}} \leq 1.$$

The left hand side has its maximum when $|x_1 - y_1| = |x_2 - y_2|$ and so we get:

$$r\sqrt{2} \leq 1.$$

Thus we can guarantee an affine transformation is contractive by making the restriction, $r_1, r_2 \in (-0.707, 0.707)$, since then the minimum contractivity factor is given by, $s = \max\{|r_1|, |r_2|\}$. However, this does not mean that all contractive affine mappings need to satisfy this condition since it makes the assumption of the worst case values for θ_1 and θ_2. For example, a mapping for which $\theta_1 = \theta_2$ has $(ab + cd) = 0$ and then $r_1, r_2 \in (-1, 1)$ satisfies the contractivity condition.

The Manhattan metric defined as $d_m(x,y) = |x_1 - y_1| + |x_2 - y_2|$ is a less expensive function to evaluate than the Euclidean equivalent and so it would appear good sense to adopt this as our working metric. However, if we calculate the contractivity factor for an affine mapping under the Manhattan metric we arrive at the following:

$$d_m(w(x), w(y)) = (|a| + |c|)|(x_1 - y_1)| + (|b| + |d|)|(x_2 - y_2)|.$$

Now,

$$|a| + |c| = |r_1||cos\theta_1| + |r_1||sin\theta_1| \leq |r_1|\sqrt{2},$$

and similarly $(|b| + |d|) \leq |r_2|\sqrt{2}$. With $r = \max\{|r_1|, |r_2|\}$ we get:

$$d_m(w(x), w(y)) \leq r\sqrt{2}\ d_m(x,y).$$

Hence, to ensure contractivity we have the requirement that $r_1, r_2 \in (-0.707, 0.707)$ for all mappings. Therefore the choice of metric determines the range of scale factors that can be used and hence the range of collages that can be made. In order to maximise the chances of obtaining a good collage, we decide to use the Euclidean metric and the larger range of scale factors it makes possible. Also, since $0 \leq cosx, sinx \leq 1$ for all x, we have a, b, c, d strictly less than unity and with normalised coordinates we have all the six mapping coefficients less than unity as promised earlier. The full significance of the limits on mapping scale factors will become apparent in chapter six where we are concerned with generating random contraction mappings.

An additional property of two-dimensional affine mappings that makes them useful for the current implementation is that the area which they cover is easily calculable from their coefficients. This is of relevance to the random iteration algorithm for which it has been stated that the probabilities associated with each mapping should be in proportion to their area. If the area of a shape is $\mathcal{A}$ then

the area of that shape under an affine transformation is given by $|ad-bc|\mathcal{A}$. Thus we take the probability for each mapping as:

$$P_n = \frac{|a_n d_n - b_n c_n|}{\sum_{i=1}^{N} |a_i d_i - b_i c_i|}.$$

Of course, this solution is only strictly applicable to totally disconnected or just-touching iterated function systems, but can still be applied to slightly overlapping ones without a significant degradation in the distribution of points over the attractor. In any case, overlapping codes are inefficient from the information storage point of view and so we will normally be looking for totally disconnected or just touching representations and the overlap problem will not become apparent. The only care that need be exercised with such probability assignments is that each value of P_n does not fall below the minimum non-zero number that can be represented in the system. If a probability is ever set to zero the associated map is effectively removed from the IFS and the representation is degraded as a consequence.

4.3 Encoding Algorithm

We now give a detailed description of the proposed collage constructing algorithm. The data on which it works takes the form of digitised binary images of simple geometric shapes contained within a 512 by 512 pixel array. To begin we give an overview of the structure of the algorithm.

1. The boundary points of the input shape are detected using a simple edge tracking algorithm. An arclength value is assigned to each point on the boundary and the centroid of the shape is calculated. (The arclength value for a point is its linear distance along the boundary from a chosen origin).

2. The curvature, κ, is calculated at each boundary point by the convolution of the raw edge data with derivatives of a Gaussian function, $G(\sigma, s)$, where s is the arclength parameter.

3. The boundary is segmented into a number of arcs, the endpoints of which correspond to zero crossings of curvature.

4. Any arcs of less than five pixels in length are treated as spurious and merged with surrounding arcs.

5. The equation of an interpolating arc, parameterised in terms of arclength, is calculated using a least squares method. Each arc is classified as either linear, concave, or convex by inspection of the equation coefficients, and then placed in a data queue.

6. Contractive affine transformations are calculated which map arcs of the same curvature type onto each other. Only mappings with contractivity factors less than 0.8 are considered so as to avoid the value of $(1-s)^{-1}$ becoming too large. The quality of each generated mapping is tested by evaluating an error function given by:

$$E = E_a \sqrt{\frac{(r_1^2 + r_2^2)}{2}};$$

where E_a is a measure of the area of the mapping that does not overlay the shape, and r_1 and r_2 are the mapping parameters introduced earlier. All error values are normalised to the range $[0, 100]$.

7. If the best error value obtained for mappings to a given arc is less than a set threshold, or if the arc becomes shorter than one pixel, then the corresponding mapping is accepted as part of the IFS and is output to a file. The matched arc is then removed from further consideration. If no suitable match is found then

the unmatched arc is halved and the two new arcs produced are added to the back of the queue.

8. The search continues, considering each arc in the queue in turn, until the queue is empty.

Boundary detection is achieved by an edge tracking algorithm that works by finding points (pixels) adjacent to a known starting edge point. It makes the assumption that the boundary it is following is continuous, closed, and relatively smooth. It makes its traversal in an anticlockwise direction, (so determining the direction of increasing arclength, and thus giving meaning to the terms concave and convex), so that it moves in the sense of increasing angle subtended at the centroid, as usually reckoned in mathematics. The eight possible directions in which an adjacent point may lie are labelled as in the following table.

0	1	2
7	$\star$	3
6	5	4

Table 4.1 *The possible search directions relative to the pixel marked* $\star$ *as used by the boundary tracking algorithm.*

We make the assumption that the shape to be coded is contained within a known rectangular area of interest (AOI), and is of such a size that it intercepts the main diagonal. (This is equivalent to knowing the extent of the shape in the x- and y-directions which in any case is necessary for putting the bounds on the translation component of mappings). Starting from the top left-hand corner of the AOI, the algorithm samples each successive point along the leading diagonal until it detects one that is part of the shape - a 'foreground' point. This is taken as the first boundary point and is assigned a zero arclength value. The

search direction at which this point was found is taken as the step direction from the previous 'background' point as defined in table 4.1. The eight surrounding points are then checked in an anticlockwise direction, starting with the one in the position 'most opposite' (see later) to the search direction. When one of these points is detected as also being part of the foreground, it is taken as the second boundary point. With the location of this second point it is possible to make an assumption, based on the supposed smoothness of the boundary, as to the local direction in which the boundary is progressing, and so the following routine can be used for the rest of its points.

From the 'old' search direction, the 'new' search direction is calculated using the formula $new = (old + 5) \pmod 8$, which is the definition of the 'most opposite' direction. Starting in this direction each surrounding point is checked in a clockwise direction until a background point is detected. The last foreground point before this one is taken as the next boundary point, its step direction is taken as the 'old' direction from which the new one is calculated. Repeating this process tracks the remainder of the boundary. The algorithm terminates when the last found point is adjacent to the initial point, so long as the boundary contains more than three points.

As an example consider the section of pixel array represented below where the shaded points represent the shape (foreground). The search starts at the point in the top left-hand corner of the array. Moving in direction 4 the fifth point it checks (a) is determined to lie on the boundary and is taken as the initial point. The new search direction is set as 7 and so, starting with the pixel in direction 7 and moving anticlockwise, (decreasing numerical search directions), each point adjacent to (a) is tested. The first such point tried is (b) and it is also found to be part of the foreground and is taken as the second boundary point. The new

search direction is calculated to be $4 = (7 + 5) \pmod 8$. Searching clockwise from (b) and starting in direction 4, the last foreground point is found to be (c). The new starting search direction yields (d) as the next boundary point. Continuing in this fashion, the tracking stops when it reaches the point (z).

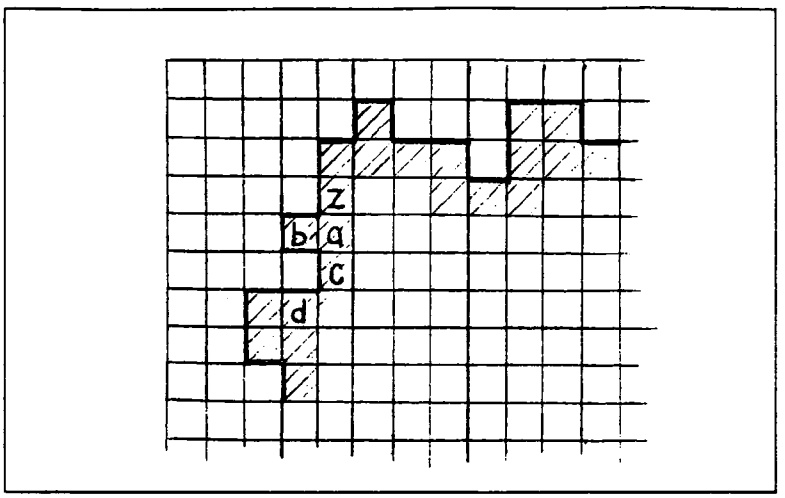


Figure 4.2 *An example of the operation of the boundary tracking algorithm.*

For each boundary point the approximate arc-length distance from the preceding one is calculated and stored. Points in the step directions 0, 2, 4, and 6 are taken as being $\sqrt{2}$ units away, whilst those in directions 1, 3, 5, and 7 are one unit away. From this information we can associate an arc-length value, s, to each point along the boundary.

The required end result of the segmentation process is a set of boundary arcs between which it will be possible to find contractive mappings. We refer to this mapping process as 'matching' one arc to another. Clearly better matches are possible between arcs of the same structure - ie. smooth or angular - and so we choose to segment at points of curvature zero crossings. This means that along its length each arc will have either zero curvature, or else a non-zero curvature

of a constant sign. The concept of segmentation on curvature values is widely accepted because of the high perceptual information associated with points of high curvature - see for example Asada and Brady (Asada and Brady 1986).

Curvature, κ, is defined as the rate of change of a curve's gradient with arclength. That is:

$$\kappa = \frac{d\phi}{ds} \qquad \text{with} \qquad \tan\phi = \frac{dy}{dx}.$$

Following the boundary detection phase, we have an arclength value associated with each point (x, y), and so it is natural to express curvature in terms of the derivatives of x and y with respect to s. This is the same approach as that taken by Mokhtarian and Mackworth (Mokhtarian and Mackworth 1986). We begin with the expression for ϕ and differentiate both sides with respect to s.

$$\frac{d(\tan\phi)}{ds} = \sec^2\phi\frac{d\phi}{ds} = \frac{d}{ds}\left(\frac{dy}{dx}\right).$$

Making the substitutions $\dot{x} = dx/ds$, $\dot{y} = dy/ds$, $\ddot{x} = d^2x/ds^2$ and $\ddot{y} = d^2y/ds^2$, and expressing dy/dx in terms of $\dot{x}$ and $\dot{y}$, we obtain:

$$\kappa = \frac{d\phi}{ds} = \cos^2\phi\frac{d}{ds}\left(\frac{\dot{y}}{\dot{x}}\right).$$

Expanding the differential we get,

$$\frac{d}{ds}\left(\frac{\dot{y}}{\dot{x}}\right) = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\dot{x}^2},$$

and substituting for the $\cos\phi$ term using:

$$\cos^2\phi = \left(1 + (\dot{y}/\dot{x})^2\right)^{-1} = \frac{\dot{x}^2}{\dot{x}^2 + \dot{y}^2},$$

we arrive at the result:

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}.$$

This differs from the result that Mokhtarian and Mackworth obtained, in that they have the denominator raised to a power of 3/2. We claim that our result is

the correct one, arguing that the introduction of a fractional power introduces an ambiguity as to the sign of the curvature. However, the error is unimportant with respect to Mokhtarian and Mackworth's work on scale-space representation since the denominator is dimensionless and their erroneous values are only scalings of the correct ones.

Due to the discrete nature of a digitised image, and the effects of noise, calculation of κ based on the raw data would contain too many spurious zero crossings to be of any use. Hence we smooth the data by convolution with a Gaussian function defined by,

$$G(\sigma, s) = \frac{1}{\sigma\sqrt{2\pi}} exp(-s^2/2\sigma^2),$$

and make the substitutions:

$$\dot{X} = x(s) * \frac{dG(\sigma, s)}{ds} \qquad \ddot{X} = x(s) * \frac{d^2G(\sigma, s)}{ds^2};$$

with $\dot{Y}$ and $\ddot{Y}$ defined similarly.

The boundary is now segmented into a number of arcs, the endpoints of which correspond to curvature zero crossings. The next stage of the algorithm is to find a parameterised equation for each of these arcs which requires that an arc contain at least three boundary points. Therefore to ensure a meaningful amount of data in each arc, and to weed out any spurious ones that have evaded the smoothing filter, arcs of less than five boundary points in length are merged with surrounding arcs.

Each arc is represented as a quadratic function of s since it is known that each arc has a constant curvature sign and, by inspection of the curvature equation just derived, so must such a parameterisation. Explicitly we make the assignments:

$$x(s) = a_2s^2 + a_1s + a_0, \qquad y(s) = b_2s^2 + b_1s + b_0.$$

From which we obtain $\dot{x} = 2a_2 s + a_1$, $\dot{y} = 2b_2 s + b_1$, $\ddot{x} = 2a_2$ and $\ddot{y} = 2b_2$, and so the sign of curvature is given by:

$$sign(\kappa) = sign(a_1 b_2 - a_2 b_1).$$

Hence each arc is classified as either linear, concave, or convex, depending on whether its curvature is zero, negative, or positive. This labeling convention is determined by the choice of arclength increasing in an anticlockwise direction around the boundary. The coefficients $a_0, a_1, a_2, b_0, b_1, b_2$ are found using a least squares method. That is, for each arc the sums:

$$\Delta_1 = \sum_{i=1}^{i=N} (x_i - a_2 s_i^2 - a_1 s_i - a_0)^2; \qquad \Delta_2 = \sum_{i=1}^{i=N} (y_i - b_2 s_i^2 - b_1 s_i - b_0)^2;$$

are minimised where (x_i, y_i) are data points on the arc for $i = 1, 2, \ldots, N$. The solution to the minimisation of Δ_1 is given by the following matrix equation:

$$\begin{pmatrix} \sum s^4 & \sum s^3 & \sum s^2 \\ \sum s^3 & \sum s^2 & \sum s \\ \sum s^2 & \sum s & N \end{pmatrix}^{-1} \begin{pmatrix} \sum x s^2 \\ \sum x s \\ \sum x \end{pmatrix} = \begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

where subscripts have been dropped for clarity. The solution for Δ_2 is obtained by the same equation with the replacement of x by y throughout.

The matching of two arcs is achieved by calculating a number of points along the length of each and finding the best fit transformation that maps one set of points onto the other. This is a further reason for parameterising everything in terms of s since any number, m, of points may be chosen along the length of each arc depending only on the resolution required. If the points along the first arc are denoted by the set of coordinates $(u(s), v(s))$, and those along the second arc by $(x(s), y(s))$, then the required mapping, w, is given by,

$$u_n = a x_n + b y_n + e \qquad \text{and} \qquad v_n = c x_n + d y_n + f,$$

and is found by the minimisation of

$$\Delta_3 = \sum_{n=1}^{m}(u_n - ax_n - by_n - e)^2 \quad \text{and} \quad \Delta_4 = \sum_{i=n}^{m}(v_n - cx_n - dy_n - f)^2.$$

The following matrix equation (again without subscripts) gives the solution for (a, b, e) whilst the replacement of u by v gives (c, d, f).

$$\begin{pmatrix} \sum x^2 & \sum xy & \sum x \\ \sum xy & \sum y^2 & \sum y \\ \sum x & \sum y & m \end{pmatrix}^{-1} \begin{pmatrix} \sum xu \\ \sum yu \\ \sum x \end{pmatrix} = \begin{pmatrix} a \\ b \\ e \end{pmatrix}.$$

The quality of a given mapping is decided by an error function defined by:

$$E = E_a \sqrt{\frac{(r_1^2 + r_2^2)}{2}}.$$

where r_1 and r_2 are the scaling coefficients given by $\sqrt{a^2 + c^2}$ and $\sqrt{b^2 + d^2}$ respectively, and E_a is the fraction of points in the whole of the shape that get mapped onto the background. A good mapping is one which correspond to a low error value. There are several reasons for the use of this form of error function over the others that immediately present themselves. Firstly, it is not enough to use the values of Δ_3 and Δ_4 as the error measure since they only relates to how well two arcs match each other and do not give any information as to how good the mapping is as part of a collage. There is not even any reason why their values should be used as an element of the error function since it is only required that the mapping produced be the best possible (for a least squares fit) between two given arcs - it is only the possibility of a match between the arcs that is of concern. Secondly, the Hausdorff distance is not included in the error measure for two reasons. The most important is that it is only of use when applied to the difference between the shape and the whole of the collage. It is possible that the Hausdorff distance between a shape and a single mapping could be quite large but that the mapping still be a valuable component of a collage. Also, by concentrating collage construction around the boundary of a shape we

are resigned to the fact that our final collage may be a poor representation of the whole shape, and that there will be a large Hausdorff distance between the two. Thirdly, the evaluation of E_a determines not only how good the match is at the site of interest, but also globally. It will penalise mappings that produce overspill of the shape at other points of the boundary other than that currently being considered.

The use of the Pythagorean sum of scale factors mitigates the effect of overspill by scaling it to the 'size' of the mapping. For this factor to be small it requires that both r_1 and r_2 be independently small, and thus avoids the acceptance of long thin mappings which may have a very small area but which protrude from the shape and cause the associated attractor to form 'whiskers'. The value of E_a is scaled so that possible error values are in the range $[0, 100]$. The overall effect of the error function is to favour mappings that maximise the number of points that they map onto the shape (the foreground) whilst minimising the number they map onto the background.

Matchings are not considered between all possible pairs of arcs, in fact each arc is only tested for a match with those of the 'matching set'. This is the set of arcs accepted after the initial segmentation of the boundary. This restriction is employed to prevent a rapid increase in the size of the search space, and also to guarantee that most of the possible matchings will be contractive. Further, only matches between arcs of the same type are considered in order to maximise the possibility of a good fit. The order in which arcs are tested is determined by their position in the arc queue. At the start each of the arcs from the boundary segmentation is placed in the queue in an arbitrary order. The first arc is then considered, and a match looked for with arcs of the same curvature type in the matching set. If the error value for the best mapping for a given arc is less

than the set threshold value, or if the length of the arc has become less than one pixel, the mapping will be accepted as part of the IFS, and the arc will be deleted from the queue. However, if no suitable mapping is found then the arc is halved to produce two child segments which are placed at the back of the queue. The next arc in the queue is then taken, and the process continues until the queue is empty.

When an arc is halved the equations and types of the two new arcs are recalculated if they are of a length greater than five pixels, otherwise they inherit the equation and type of their parent. This ensures that the best possible equation is used for each arc and that arcs that are too short for the least squares matching process still have a meaningful equation. We can be sure that the algorithm will terminate since successive halvings of arcs ensures that their lengths must eventually become less than one pixel at which point the best mapping is accepted regardless of the error threshold.

Finally we demonstrate that the algorithm has an upper bound of $O(n^2)$ where n is the number of boundary points. Consider the segmentation of the boundary to result in m arcs each of length $a_i n$ for $i = 1, 2, \ldots, m$ such that:

$$\sum_{i=1}^{i=m} a_i = 1.$$

An arc of length $a_i n$ can be halved until each child is less than or equal to one pixel in length. Thus an upper bound on the number of times an arc can be halved is $h = \log_2(2a_i n)$ and so an upper bound on the number of children it can produce is given by:

$$\sum_{j=0}^{j=h} 2^j = 2^{h+1} - 1 = 4a_i n - 1.$$

Neglecting overheads, the execution time of the algorithm is proportional to the the total number of matches it considers. In the worst case every arc produced

is matched to each arc in the matching set and so we have the total number of matches given by:

$$\sum_{i=1}^{i=m} m(4a_i n - 1) = 4mn \sum_{i=1}^{i=m} a_i - m^2 = 4mn - m^2.$$

However, we restrict the minimum length of an initial arc to five pixels, and so the maximum value of m is $n/5$. Hence an upper bound on the number of matches made is $19n^2/25$ and so the algorithm is at worst $O(n^2)$.

4.4 Program Performance

We now present the results produced by a program of the algorithm described in the previous section. To test the program's performance we used a set of four simple shapes containing different combinations of the three arc types. The test set is shown in figure 4.1 and, labeling from left to right, we shall refer to them as shapes one, two, three, and four respectively.

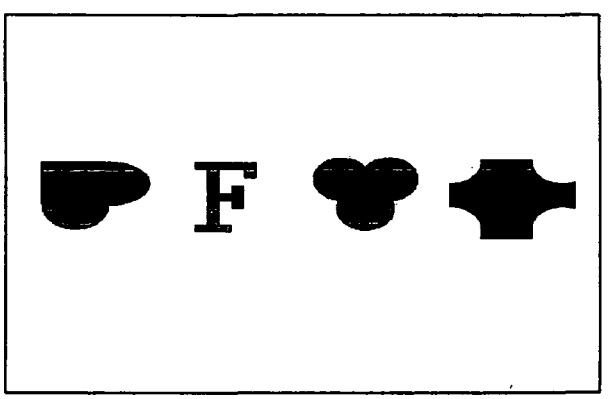


Figure 4.3 *The set of test shapes used to test the performance of the encoding algorithm.*

Whilst quantitative results are supplied for encoding speeds and compression ratios, the fidelity of the representations are discussed only qualitatively. This is

due to the problem inherent in calculating perceptually meaningful values for the 'difference' between two shapes. Mumford (Mumford 1987) discusses this problem and gives examples of shapes which demonstrate that the use of metrics such as the Hausdorff distance quantify the 'closeness' of two shapes in a way that does not in general correspond to human classification, indicating that the shape of an object is a complex and context dependent perceptual entity. We therefore make the assumption that a good encoding is one that appears of high quality to the human eye and base our assessment of program performance upon this evaluation.

We start with the output from the boundary detection and segmentation routine which is dependent on the smoothing parameter σ. The effect on the number and position of the arcs into which the boundary is divided as σ is increased is demonstrated for shape two by the sequence of pictures in figure 4.4.

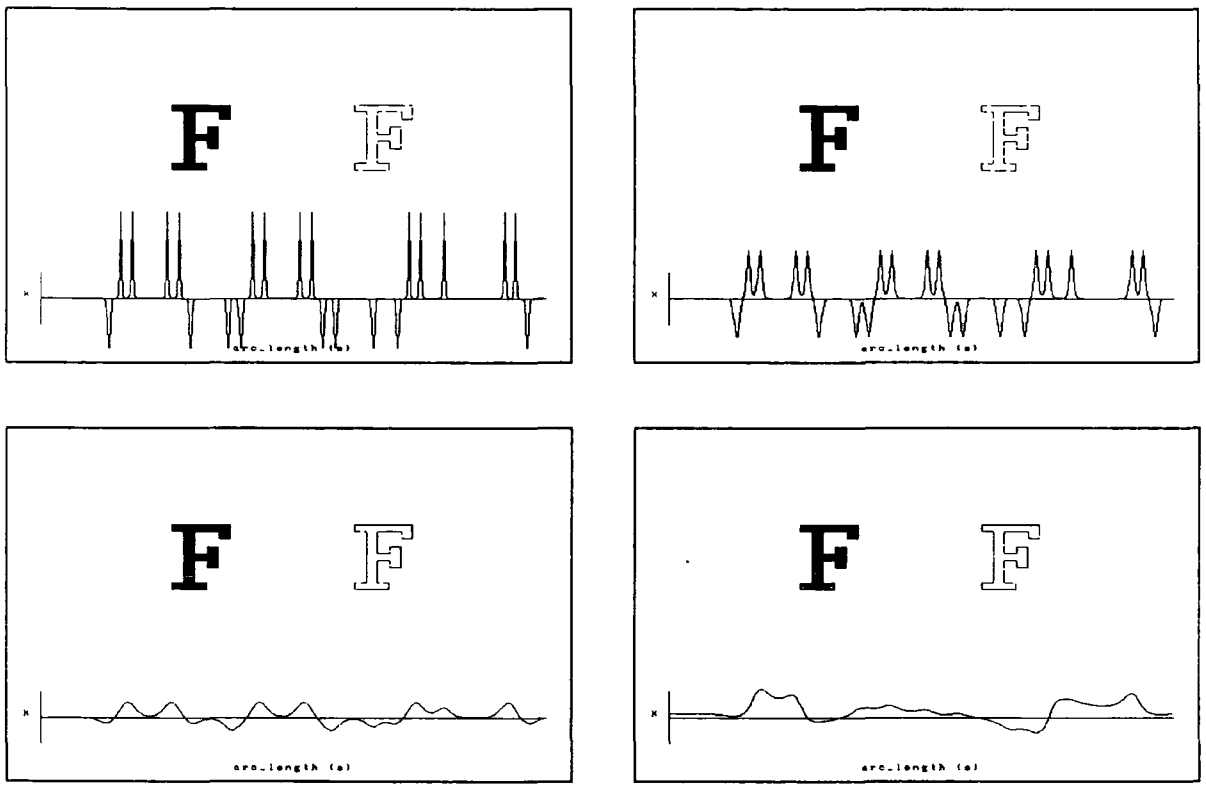


Figure 4.4 *A sequence of boundary segmentations of shape two for increasing values of σ. Arc endpoints are represented by the breaks in the shape outline. The values of the smoothing parameter are: (top left)* 1.0, *(top right)* 2.2, *(bottom left)* 8.0, *and (bottom right)* 32.0. *For each picture the graph at the bottom is the curvature plot on which the segmentation is based.*

Clearly, for an angular shape such as that used, the boundary of which could be accurately described using only linear arcs, a low value of σ is required to obtain an optimal segmentation since the high sharp peaks produced at the corners maximise the length of the linear sections. However, as described in the previous section, in order for the program to have some degree of robustness when used on noisy images, any arcs of less than five pixels in length are treated as spurious and are merged with surrounding arcs. Hence, as the results for $\sigma = 1.0$

demonstrate, if the smoothing parameter is made too small then the curvature peaks at corners become so sharp that the small concave and convex arcs that they represent fall below the minimum length and cause linear arcs to be joined. Since the typing of each arc is left until its parameterised equation has been calculated, linear arcs joined in this way become classified as curves. The best segmentation of shape 2 is achieved with a smoothing parameter of $\sigma = 2.2$, the broader curvature peaks being accepted as individual arcs, and thus allowing the correct classification of the whole of the boundary. As σ is increased further, the peaks become still broader resulting in the shortening, and in some cases the disappearance of, the linear segments. This is demonstrated by the results for $\sigma = 8.0$ where it can be seen that some of the shorter linear segments detected with smaller smoothing parameters have joined with adjacent curves due to the coalescence of curvature peaks, as clearly visible in the graph. For $\sigma = 32.0$ as shown in the final picture of the sequence, the curvature plot is smoothed to such a degree that the boundary is segmented into only four arcs, resulting in the loss of all fine detail of the boundaries structure and its subsequent representation as four long curves. Although not shown, if the smoothing is continued any further curvature becomes positive along the boundary's entire length and no segmentation occurs.

It was determined experimentally that a value of $\sigma = 2.2$ gave useful segmentations of all the shapes in the test set. This is demonstrated by the following figure.

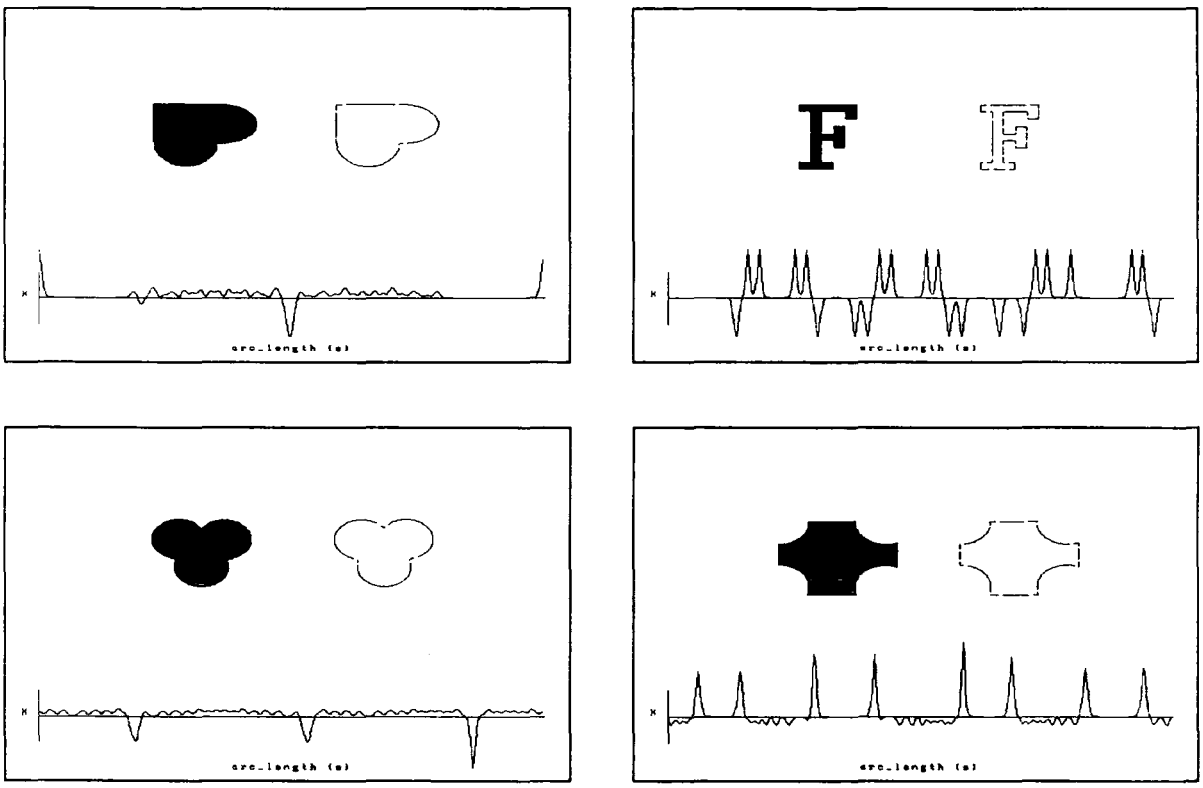


Figure 4.5 *Boundary detection and segmentation for the set of test shapes with a smoothing parameter of,* $\sigma = 2.2$. *The input shape appears on the left of each picture, and the segmentation points are denoted by the breaks in the boundaries on the right.*

The segmentations found for each test shape in figure 4.5 were used to define the initial arcs for the collage construction process. At this stage we have the error threshold, ϵ, as the free parameter and demonstrate its effect on the quality of the output collages by the following figure.

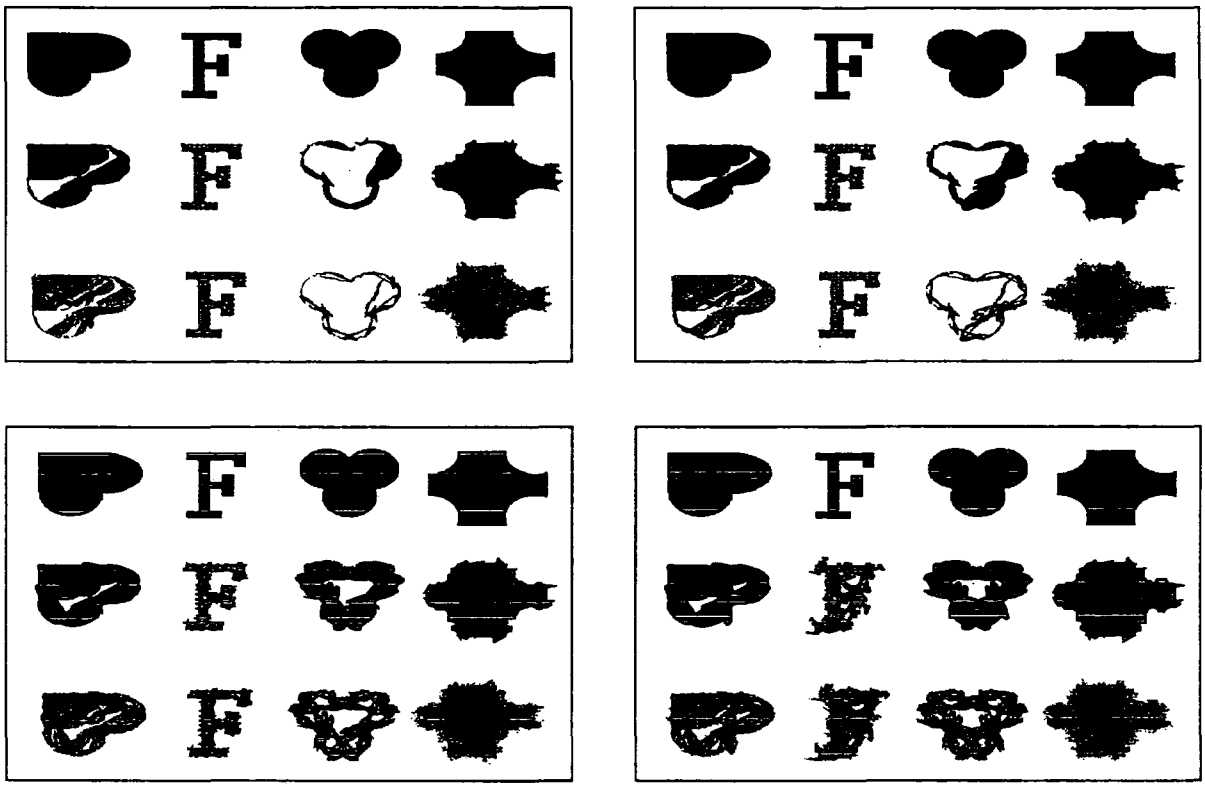


Figure 4.6 *The effects on collage construction due to increasing the error threshold, ϵ. The values of ϵ are: (top left)* 1.0, *(top right)* 2.0, *(bottom left)* 4.0, *and (bottom right)* 16.0. *For each picture the four test shapes appear on the top row with their collages directly beneath them. The collages are represented differently from usual, being shown as mappings of the whole shape to give a better idea of their space filling qualities. The bottom row of each picture shows the attractors for the IFSs associated with each collage.*

In general it appears that lower error thresholds result in better collages with more mappings (table 4.2) as is to be expected from theory. However, there are some exceptions, notably cases where an increase in the error threshold produces a better collage. First, consider the results for shape two which closely follow expectations. With the lowest threshold setting the collage consists of 138 mappings which match the boundary well and coincidentally fill the interior of the shape.

Hence the attractor of the IFS not only has a good boundary, but serves as an acceptable representation of the whole shape. For $\epsilon = 2$ the number of mappings used drops to 103 and the resulting collage fits less well to the boundary, although the interior is still well filled. Further increases result in progressive deterioration of collage quality until the attractor for $\epsilon = 32$ is unrecognizable.

ϵ	shape 1	shape 2	shape 3	shape 4
1	19	138	74	63
2	15	103	34	41
4	13	73	15	31
16	11	43	12	28

Table 4.2 *The number of mappings used to construct a collage for each shape at a given error threshold.*

The other three test shapes each show a departure from the expected norm. With shape four as input, the output is poor even for low error threshold values, with numerous 'whiskers' present in the collage and hence the attractor. This is due partly to the inherent difficulty of matching sharply pointed corners, and partly to the algorithm used. In an attempt to find mappings with below-threshold error measures, boundary arcs in the region of sharp corners are repeatedly halved. Since matching in such regions is difficult due to the narrow angle into which the shape must be mapped, this process continues until the one pixel arc length limit is reached, at which point the best mapping is accepted irrespective of its absolute error value. It is therefore possible for high error mappings to be accepted into the collage and to produce the thin peninsulas that are so apparent. A positive aspect of the collages produced for shape four is the degree to which the interior of the shape has been filled, even at low

error thresholds. However, this is due to a fortuitous combination of the geometry of the shape and the relative length of the linear arcs which enable the use of some large mappings with very low error values.

Shape one begins to display the same effect since the boundary of the collage for $\epsilon = 1.0$ is slightly inferior to that obtained with $\epsilon = 2.0$. If we take into account the quality of the collage in the interior of the shape, then both can be considered inferior to the result obtained using a threshold of 4.0 since, for a small trade-off of boundary fidelity, it achieves greater overall shape coverage. This is simply due to the use of larger mappings which naturally occupy a greater area.

Finally, shape three demonstrates several unwanted effects of the current implementation. The most noticeable is the 'migration' of collages towards the boundary as the error threshold decreases. This is not altogether unexpected since by looking only for mappings along the boundary it must be accepted that when the mappings become very small they will provide little coverage of the interior. However, in this case the effect is exacerbated by the symmetry of the shape which has the effect of producing only two distinct arcs in the matching set – one corresponding to a long convex section, and the other to a short concave one. The resulting lack of variation in the types of possible mapping, combined with the similarity of all the child arcs produced by halving, means that if no acceptable mapping can be found for an arc of given length, then it is unlikely that one can be found for another arc of similar length. In practice, this means that the program will keep searching until all the arcs get small enough for an acceptable mapping to be found. Hence for a low error threshold, nearly all the mappings are very small and the boundary of the attractor approaches one pixel in width. This is apparent in the collage for $\epsilon = 1.0$ where there is only one mapping of significant size. As the error threshold is increased further large

transforms are accepted but at the expense of degrading the boundary information. A further effect of the symmetry of shape three is the spurious structure that is apparent in the attractors of the iterated function systems produced by error thresholds of 4.0 and 16.0, which would prove misleading if used as a representation of the shape.

By selection of different (σ, ϵ) parameters for each test shape, it was possible to improve upon the results of figure 4.6. The following figure depicts the best collage that was found for each shape (using the parallelogram representation scheme) and, to demonstrate the fidelity of the information encoded in each IFS, the boundary of the associated attractor.

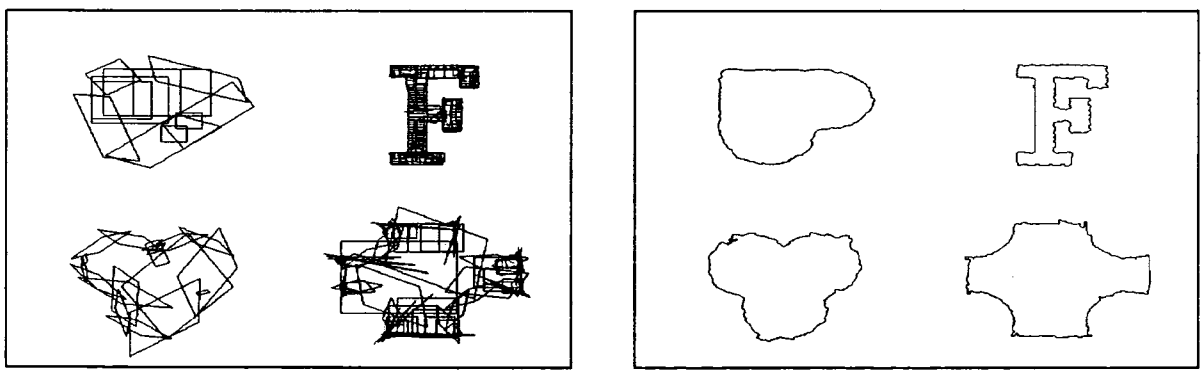


Figure 4.7 *The collages (left) and the attractor boundaries (right) produced for the best combination of parameters for each shape. The parameters are* $(4.0, 4.0)$ *for shape one,* $(2.2, 1.0)$ *for shape two,* $(1.5, 2.25)$ *for shape three, and* $(1.0, 0.5)$ *for shape 4.*

To the eye, the quality of these 'best' encodings reflects the problems specific to each shape, that of shape two for example, is of a high standard but the price we pay is the use of 138 mappings. The representation of shape one was achieved with the use of just 11 mappings but the boundary contains several significant imperfections. The encodings of shapes three and four required 19 and

97 mappings respectively, but also contain significant boundary errors. However, making the assumption that a conventional representation of shape boundaries would need to make explicit the coordinates of each point and thus require the storage of two numbers per pixel, the following table shows the compression ratios that were obtained. The figures were calculated based on the number of pixels in each shape boundary and the number of mappings used in the encodings, each of which is taken to be described by six coefficients.

shape number	boundary length	number of maps	compression ratio
1	306	11	0.108
2	449	138	0.922
3	297	27	0.273
4	383	63	0.493

Table 4.3 *Compression ratios for the best encoding of each test shape.*

To complete the results, we give the following table of approximate execution times using the parameters (2.2, 16.0) for a program running on a SUN 3/75 workstation. Clearly, as the number of mappings becomes very small as in the case for shapes one and three, our assumption that program overheads can be neglected is invalidated. The output of the RIA implementation is demonstrated by all the attractors that appear throughout this work, which were rendered at a rate of approximately one thousand pixels per second (one one-thousandth the rate achieved by Horn (Horn 1989) using an AMT DAP machine with 1024 processing elements).

shape number	boundary length	number of arcs	number of maps	runtime (cpu s)
1	306	6	11	62
2	449	18	43	354
3	297	6	12	84
4	383	16	28	288

Table 4.4 *Program execution times for* $\sigma = 2.2$ *and* $\epsilon = 16.0$.

4.5 Conclusions

We have demonstrated that automatically generated boundary collages are accessible using the $O(n^2)$ algorithm described in this chapter. For the set of test shapes used, runtimes were in the range of sixty seconds to half an hour depending on boundary length and complexity. Data compression ratios of nearly $10:1$ were achieved for the simpler shapes, and even the most complex produced codes more compact than a simple list of pixel coordinates. Qualitatively, the best encodings produced attractors which, whilst far from perfect, gave good approximations of shape boundaries. Taken together these results indicate that the encoding method is practical so long as absolute code fidelity and encoding times are not of critical importance. For example, the technique could find an application in computer graphics such as that initially suggested by Levy-Vehel and Gagalowicz.

The results further indicate that the plugging of holes in boundary collages to produce full shape encodings is not as simple as suggested by Levy-Vehel and Gagalowicz. The use of small mappings to produce accurate boundary matches can result in minimal interior overlap and hence holes which are not significantly

smaller than the original shape, and which have a more complex boundary. Fundamentally this returns us to the problem of finding full two-dimensional collages, which is the problem that boundary matching was employed to avoid.

With reference to the use of this implementation as a basis for a machine vision shape representation scheme we make the two following observations:

1. The algorithm is applicable only to the set of shapes possessing a well defined and relatively smooth boundary. In general however, we cannot expect an image of a real world object to have such a boundary for as Marr illustrates (Marr 1978), there exists a large set of shapes for which even though a clear 'perceptual' boundary exists, a well defined physical one does not. For example, we find it easy to draw the outline of an imaginary leafy tree whilst there is no physical boundary around a real tree to which our drawing corresponds. Clearly just because we perceive most shapes to have a well defined boundary it cannot be assumed that the low level processing of a machine vision system will also find them to have one. Even if such a boundary were to exist, it is unlikely to be as continuous or as smooth as those of the shapes in the test set, due to the presence of noise. A possible solution would be to take the convex hull (see for example Bailey and Cowles 1987) of a shape as its boundary, but even this is unnecessarily restricting the scope of the IFS representation scheme since, as was shown earlier, it should be possible to encode any given shape.

2. The whole algorithm depends critically on the smoothing factor σ and the error threshold ϵ, it having been demonstrated that each shape requires it own specific values of each of these parameters to achieve satisfactory results. The value of σ in particular has been shown to determine the number of initial arcs into which the boundary is segmented and therefore to determine the range of possible mappings.

The general conclusion that we can draw from the results of this chapter is that whilst the current implementation has possible applications in computer graphics, the limitations imposed on the type of shapes that can be encoded and the lack of robustness of the algorithm make it unsuitable for use as the basis of a general shape representation scheme. Specifically, we have shown that the problem of finding a full two-dimensional collage cannot be avoided by this boundary matching approach, and that it is necessary to develop an algorithm for directly obtaining full collages.

The construction of such an algorithm poses a complex problem. Even using a normalised representation scheme, and restricting the mapping coefficients to an accuracy of two decimal places, we have a set of mappings numbering in the region of one hundred million, from which to choose. Further, there is no known method for finding the optimal collage other than 'try all cases' (Aho et al. 1983) which is clearly impractical. Hence, as for the vision problem in general, we are forced to look for approximate solutions. Fortunately we know that a IFS does not have to be perfect to be of use and that we can easily determine the quality of a given IFS using the collage theorem.

The situation just described is the type of problem domain for which adaptive algorithms were developed. Hence we devote the next chapter to the description of a special type of adaptive algorithm, the genetic algorithm, and go on in chapter six to implement an IFS encoding scheme based upon one.

5 GENETIC ALGORITHMS

For any search algorithm intended to function in a large and complex domain, there exists a fundamental trade-off between two competing strategies, those of exploration and exploitation. That is, whether to focus attention in the direction of the locally best solutions or to perform a more exhaustive search of the whole space, regardless of local qualities. The first approach is exemplified by random search techniques such as that of 'try all cases' which, although eventually guaranteed to produce the optimal result, are too inefficient to be of any practical use (Aho et al. 1983). Hill climbing algorithms employ the second type of strategy in that they adopt the best solution currently available and as a consequence are easily trapped in local maxima, possibly leaving large areas of the search space unsampled. A feature of both random search and hill climbing algorithms is that they discard much of the information presented to them during the course of a search. It has been shown (Holland 1975) that a genetic algorithm (GA) achieves a near optimal trade-off between exploration and exploitation and also makes good use of past experience. Genetic algorithms have been demonstrated to have superior performance over hill-climbing types in some large search domains such as those of the NP class of problems (DeJong 1987), and find particular use in domains for which no theory exists to act as a guide. Recent applications have included uses in AI for machine learning (DeJong 1987), and in vision processing (Fitzpatrick and Grefenstette 1988)

A GA is a simplified model of the operation of population genetics (Dawkins 1976, 1982, Ridley 1983). in that it 'evolves' a set of successively better solutions. It makes the basic heuristic assumption that the optimum solution in a search space is to be found in a region containing a high proportion of good sub-optimal ones. With a view to the application of a GA to the solution of the inverse problem, this is clearly a sound hypothesis based on the inherent stability of the collage construction process as demonstrated in chapter three.

In outline, the basic structure and operation of a GA can be described as follows. In the same way that the complete set of chromosomes (or genotype) specifies the properties of the organism (the phenotype), and as such can be regarded as an encoding of that organism by a string of chemical bases, so a GA represents all possible solutions in a problem domain by fixed length strings of numbers. The solutions themselves need not be numeric since it is the way in which each string is interpreted that determines the solution's structure. However, a necessary constraint of the mapping from numeric string to solution is that it be one-to-one and onto. Employing such a representation, an initial set of trial solutions, called a population, is produced by the random generation of numeric strings. A performance measure, (or to use the language of population genetics, a 'fitness' measure), is calculated for each of the trial solutions in the population. Each solution is then allocated a number of 'offspring' proportional to its relative fitness value. The exact offspring numbers are scaled so that the total number is equal to the number of original solutions thus maintaining a fixed population size from generation to generation. Each of the offspring is then modified using a set of genetic operators which are designed to mimic the effects of biological gene recombination. This is achieved by the swapping of sections of the strings between pairs of solutions. The procedure which governs how many offspring are

to be allocated to each solution, and which of the set of genetic operators are to be applied, is called the reproductive plan. The new set of solutions generated by the reproductive plan then replace the parent population and the whole process is repeated a number of times with the result of a progressive increase in the average fitness of the population.

For the next few sections we turn to a description of the work of Holland (Holland 1975) on the development of the theory of GAs and formalise the above outline. We then discuss some of the problems encountered with practical implementations of GAs, and go on to present our own ideas for a reproductive plan that attempts to alleviate some of them.

5.1 A Formal Framework

Holland (Holland 1975) identifies the following four components as requirements of an adaptive system:

1. an environment of the system, E;
2. a set of structures, $\mathcal{K}$;
3. an adaptive plan, τ, which modifies the system structures;
4. a measure, μ, of the performance of each structure.

The purpose of an adaptive system is to update iteratively a subset of structures, $K \subset \mathcal{K}$, based on the information it receives from its environment, so that the average performance of individual structures $k \in K$ improves. In general, the form of the performance measure will depend on the environment, and so we should adopt Holland's notation of writing $\mu_E(k)$ to represent the performance of a structure in the environment, $E \in \mathcal{E}$, where $\mathcal{E}$ is the set of all possible environments. However, this notation becomes cumbersome when we start to talk of

the nth structure in a set at a time t, and because we normally have a fixed environment, we leave out explicit reference to it and write μ_{nt} as shorthand for $\mu_E(k_n(t))$, but bear in mind the proper meaning. Following Holland we make the definition:

Definition 5.1.1 Let $K(t)$ be the set of structures at time t. Let the environment produce a signal $I(t)$ which consists of the performance measures μ_{nt} for the structures $k_n(t) \in K(t)$, then the adaptive plan produces a new set of structures $K(t+1)$ and can be represented as a function:

$$\tau : I \times \mathcal{K} \mapsto \mathcal{K}.$$

As Holland observes, the relationship between $K(t)$ and $K(t+1)$ may not be deterministic since τ can be a stochastic process. That is, instead of constructing a unique set of structures, $K(t+1)$, from $I(t)$ and $K(t)$, a range of new sets of structures, $\{K_j\}$, is produced and a probability, P_j, associated with each one. The next set of structures to be evaluated is then selected with probability P_j.

The effect of iterative applications of τ to is to produce a sequence of successively fitter sets of structures which can be thought of as a trajectory, or path, through $\mathcal{K}$. Alternatively, the individual structures that comprise K can be thought of as a parallel set of trajectories through $\mathcal{K}$. (Parallel in the sense that they progress through $\mathcal{K}$ at the same time). The ability of the system to discriminate between structures is limited by the range of stimuli, I, that it receives from its environment and hence this range is a limiting factor on the improvement in average performance that is possible.

It is clear that a GA is an adaptive system since we can make the associations:

$$\text{environment} \equiv \text{search environment}$$
$$\text{set of structures} \equiv \text{population}$$
$$\text{structures} \equiv \text{solutions}$$
$$\text{performance measure} \equiv \text{fitness}$$
$$\text{adaptive plan} \equiv \text{reproductive plan}$$

which are consistent with the outline of a GA's operation and with the previous definitions. In future when referring to GAs we will use the terminology on the right since it is more descriptive, although we will always intend the meaning formally associated with the terms on the left. In addition we make two further terminological definitions. Firstly, we will refer to each successive population that is generated during the iteration of the algorithm as a generation, for example the initial random population is the first generation, and so on. Secondly we call the fittest solution that has been found by the nth generation the best-so-far solution. The term indicates that although the solution is the best that has been found to date, it does not preclude the possibility that a few more generations would yield a better one. The best-so-far solution is important as it is the result we are interested in from a practical point of view, the average fitness of the final generation being of incidental importance, although providing a useful measure of the algorithms performance.

The set of solutions, $K(t)$, serves a double purpose, being not only the data on which the algorithm is currently working, but also a coded history of all the structures tried to date. How this is possible is described in the following

sections, but first it requires the specification of exactly what we mean when we talk of data strings, reproductive plans, and genetic operators.

5.2 Schemata

In the analysis of Holland it was shown that best results can be expected from a GA when solutions are represented as binary strings and, since any set of solutions which can be represented by a string of numbers can obviously be represented by a binary string, it is unusual for anything else to be used. Hence, the following discussion assumes the use of binary strings in all cases.

Borrowing the terminology from genetics, the use of a binary representation means that each point (or locus) on the string can be occupied by one of only two 'alleles'. This simply means that there can be either a '1' or a '0' at each point of the string. We can then represent a section of a string as follows:

$$\ldots 1-0-0-1-1-\star-\star-\star-\star-0-1\ldots$$

where the $\star$ stands for a locus at which the allele value is of no importance to the current discussion. Holland gives the following definitions.

Definition 5.2.1 A **schema** is an n-tuple of defining positions along a binary string.

That is to say we view each solution string as a compound entity consisting of a combination of different loci groupings. The groupings are allowed to overlap and a single locus is permitted to be a member of more than one distinct schema. Thus a string of length l loci contains 2^l different possible schemata. A specific

association of allele values to the defining positions of a schema is called an instance or realisation of that schema. The previously illustrated section of a binary string can be interpreted as a instance of a schema with seven defining positions, and is just one of the 2^7 possible realisations.

Definition 5.2.2 Let a schema, ϵ, have n defining positions $i_1, i_2, i_3, \ldots, i_n$ along a binary string. The **length** of the schema is defined to be:

$$l(\epsilon) = (i_n - i_1).$$

Hence the schema of our example has a length of ten units.

Schemata are treated as the random variables in a population and as the real entities being evaluated when the fitness of a solution is calculated. For example, the observed average fitness of an instance of a schema is taken as the average fitness of all the solutions in which it appears. An analysis of the change in the relative proportions of schema instances leads to an explanation of the power of a GA, but first we must consider the way in which schemata are modified from generation to generation.

5.3 Reproductive Plans

Two possible reproductive plans are described by Holland, and are labeled R_1 and R_2 respectively. Plan R_1 updates only one solution during each generation and is defined as follows.

Reproductive Plan $\mathbf{R}_1$

1. Set $t = 0$ and initialise the population randomly.
2. Calculate and store the fitnesses, μ_{n0}, for $n = 1, 2, \ldots, N$.
3. Increment t by one.
4. Select a solution with probability, $P_i = \mu_{i(t-1)}/N\bar{\mu}_{t-1}$,
   where $\bar{\mu}_t$ is the average population fitness.
5. Apply genetic operators to produce a child solution.
6. Choose a second solution at random from the population.
7. Replace the chosen solution with the new child and calculate its fitness.
8. Repeat steps three to seven until $t = T$, where T is the runtime allocated.

The second algorithm, R_2, uses a time-step during which all the solutions are updated, and for which each individual solution is replaced deterministically.

Reproductive Plan $\mathbf{R}_2$

1. Set $t = 0$ and initialise the population randomly.
2. Calculate and store the fitnesses, μ_{n0}, for $n = 1, 2, \ldots, N$.
3. Increment t by 1.
4. For each solution, k_i, generate n_i offspring by selecting n_i choices of genetic operators with $n_i = \mu_{i(t-1)}/\bar{\mu}_{t-1}$ where $\bar{\mu}_t$ is the average fitness.
5. Place all the offspring in the next generation.
6. Replace the parent generation with the child generation.
7. Repeat steps three to six until $t = T$, where T is
   the number of generations to be run.

Based upon these reproductive plans, Holland states the following theorem.

Theorem 5.3.1 *(Holland 1975) If at any time-step, t, there is probability p_1 that a structure, $k_n(t)$, produces an offspring during that time-step, and there is a probability p_2 that $k_n(t)$ is deleted during that time-step, then the expected number of offspring of $k_n(t)$ is p_1/p_2.*

Proof – The probability of $k_n(t) \in K(t)$ surviving a time-step is $(1 - p_2)$, so the probability of $k_n(t)$ being deleted during the Tth time-step is its probability of surviving $T - 1$ time-steps multiplied by its probability of being deleted during the Tth. That is

$$p(T) = (1 - p_2)^{T-1} p_2.$$

The expected number of offspring during this interval is simply $\bar{\mu}_{nT} = p_1 T$. Therefore, the expected number of offspring during the lifespan of k_n is:

$$\begin{aligned} \sum_{t=1}^{\infty} p(t)\bar{\mu}_{nt} &= \sum_{t=1}^{\infty} p_1 p_2 t(1 - p_2)^{t-1} \\ &= p_1 p_2 \sum_{t=1}^{\infty} t(1 - p_2)^{t-1}. \end{aligned}$$

However:

$$\sum_{t=1}^{\infty} t(1 - p_2)^{t-1} = 1 + 2(1 - p_2) + 3(1 - p_2)^2 + \ldots = (1 - (1 - p_2))^{-2} = \frac{1}{p^2}.$$

and so,

$$\sum_{t=1}^{\infty} p(t)\bar{\mu}_{nt} = \frac{p_1 p_2}{p_2^2} = \frac{p_1}{p_2}.$$

For plan R_1 we have:

$$p_1 = \mu_{nt} / \sum_{m=1}^{m=N} \mu_{mt} \qquad \text{and} \qquad p_2 = \frac{1}{N} \qquad \forall\ k_n(t);$$

whilst for R_2 we have:

$$p_1 = \mu_{nt}/\bar{\mu}_t \qquad \text{and} \qquad p_2 = 1.0 \qquad \forall\ k_n(t).$$

Then, if the total fitness of the population changes negligibly over the expected lifetime, the expected number of offspring for $k_n(t)$ under either plan is:

$$p_1/p_2 = N\mu_{nt}/\sum_{m=1}^{m=N}\mu_{mt} = \mu_{nt}/\sum_{m=1}^{m=N}\frac{\mu_{mt}}{N}$$
$$= \mu_{nt}/\bar{\mu}_t.$$

where $\bar{\mu}_t$ is the mean fitness of all solutions in the population at time t.

Thus for a reproductive plan of type R, which is to say either R_1 or R_2, the number of offspring allocated to each solution is dependent on its relative fitness. Solutions with above average fitness clearly get allocated more offspring whilst those of below average fitness get fewer. Since only integer numbers of offspring are actually possible, Holland suggests scaling the value of $\mu_{nt}/\bar{\mu}_t$.

5.4 Genetic Operators

Holland describes several genetic operators such as mutation, crossover, inversion, dominance, modification, translocation, and deletion, but shows that just mutation and crossover are adequate for a robust and general purpose set of operators.

To discuss the use of these operators we represent a solution string as $k = a_1a_2a_3\ldots a_l$ where a_i denotes the allele value at each locus, and is thus either a '1' or a '0', and l is the length of the string. Holland then describes the crossover operator as the following three step process:

1. Two structures, $k = a_1 a_2 \ldots a_l$, and $k' = a'_1 a'_2 \ldots a'_l$, are selected at random from the current population.
2. A crossover point is selected by choosing a random number, x, from $\{1, 2, 3, \ldots, l-1\}$.
3. Two new structures are formed by exchanging the alleles of k and k' to the right of position x which results in the new structures:

$$a_1 a_2 \ldots a_x a'_{x+1} \ldots a'_l, \qquad a'_1 a'_2 \ldots a'_x a_{x+1} \ldots a_l.$$

The effect of a crossover operation on the schemata pool is twofold. Firstly there is the generation of new instances of a schema already in the pool. For example, the structure k given above is an instance of the schema $a_1 a_2 \ldots \star\star\ldots\star$ but after crossover with k' we get a new instance of $a_1 a_2 \ldots \star\star\ldots\star$ namely that of $a_1 a_2 \ldots a_x a'_{x+1} \ldots a'_l$, always assuming that $a_i \neq a'_i$ for some $i > x$. Each new instance of a schema ϵ is equivalent to another trial of the random variable associated with ϵ, and so increases the the likelyhood that the observed mean performance $\bar{\mu}_{\epsilon t}$ of the schema ϵ is a good approximation of the expectation of the random variable ϵ. Crossover also generates completely new schemata for trial. The crossover between k and k' produces an instance of the schema $\star\ldots\star a_x a'_{x+1} \star\ldots\star$ which was present in neither of the two initial strings, again assuming that $a_{x+1} \neq a'_{x+1}$. Holland gives the following theorem for the number of instances of new and already existing schemata that are generated by a crossover operation.

Theorem 5.4.1 *(Holland 1975) Let $k = a_1 a_2 \ldots a_l$, and $k' = a'_1 a'_2 \ldots a'_l$, differ in attribute values at n_1 positions to the left of $x+1$ and by n_2 positions to the right. Then either resultant of crossover between k and k' will be an instance of*

$2^l - 2^{l-n_1} - 2^{l-n_2} + 2^{l-n_1-n_2}$ *'new' schemata and an instance of* $2^{l-n_1} + 2^{l-n_2} - 2^{l-n_1-n_2}$ *schemata already instanced by* k *or* k'*, assuming* $n_1 \neq 0$ *and* $n_2 \neq 0$.

Proof – After crossover any schema which is defined at one or more of the n_1 positions to the left, and at one or more of the n_2 positions to the right, will have neither k nor k' as an instance. The number of such schemata is the number of 'new' schemata instanced by the crossover. There are 2^{n_1} combinations that can be made of the n_1 positions to the left. However, this includes the 'null' combination in which none of the n_1 positions are defined and so the number of combinations including at least one positions is $2^{n_1} - 1$. Similarly to the right there are $2^{n_2} - 1$ combinations. The remaining $l - (n_1 + n_2)$ positions can have any allele and so there are $2^{l-n_1-n_2}$ combinations possible. The total number of 'new' schemata is then:

$$(2^{n_1} - 1)(2^{n_2} - 1)(2^{l-n_1-n_2})$$

$$= 2^l - 2^{l-n_1} - 2^{l-n_2} + 2^{l-n_1-n_2}.$$

Since n_1,$n_2 > 0$ the remainder of the 2^l schemata will have new instances. Thus the number of existing schemata getting new instances will be:

$$2^{l-n_1} + 2^{l-n_2} - 2^{l-n_1-n_2}.$$

Clearly the application of the crossover operator allocates new trials to existing schemata whilst simultaneously introducing new schemata for trial. This is where the GA manages the trade-off between exploitation and exploration since further trials of existing schemata represent a more thorough examination of regions of the search space already identified as containing above average solutions, whilst the new schemata represent a speculative investigation of as yet unexplored areas.

We now consider the second of Holland's genetic operators, that of mutation. This is a single step process for each loci of a string whereby the allele value is inverted with probability P_m. For example, if the binary string of structure k begins with the sequence,

$$1-0-1-1-0-\ldots,$$

then a mutation at the third locus will transform it to:

$$1-0-0-1-0-\ldots.$$

The mutation probability, P_m, is the same for each locus and has a fixed value. The purpose of mutation is to ensure that no schemata are permanently lost from the pool and is usually kept as a background process in that the value of P_m is made very small.

5.5 Intrinsic Parallelism

We are now in a position to give Holland's analysis of the operation of the reproductive plans described earlier using just the two genetic operators. This will lead to the concept of 'intrinsic parallelism' which is responsible for the effectiveness of GAs. To begin, Holland concentrates on the effect of crossover alone.

Theorem 5.5.1 *(Holland 1975) Let $P(\epsilon,t)$ be the probability of a given solution $k_n(t)$ being an instance of schema ϵ at time t. Given a reproductive plan of type R, and using only the crossover operator, the expected change in $P(\epsilon,t)$ over one generation is bounded by:*

$$P(\epsilon,t+1) \geq \left(1-\frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right)\frac{\bar{\mu}_{\epsilon t}}{\bar{\mu}_t}P(\epsilon,t);$$

where P_c is the probability of individuals undergoing crossover during a generation, and $\bar{\mu}_t$ is the observed average performance of all the schemata in the population.

Proof – It has already been shown that during one generation each individual in a population is expected to produce $\mu_{nt}/\bar{\mu}_t$ offspring under a reproductive plan of type R. Let $B_\epsilon(t)$ be the set of individuals that are instances of the schema ϵ. The total number of offspring expected for the set $B_\epsilon(t)$ is:

$$N_\epsilon(t+1) = \sum_{n:k_n(t)\in B_\epsilon(t)} \frac{\mu_{nt}}{\bar{\mu}_t} = \frac{\bar{\mu}_{\epsilon t}}{\bar{\mu}_t} N_\epsilon(t).$$

where $N_\epsilon(t)$ is the number of instances of schema ϵ at time t and $\bar{\mu}_{\epsilon t}$ is the average performance of all individuals that are instances of ϵ. If P_c is the probability of crossover selection, and $l(\epsilon)$ is the length of ϵ, then a proportion $P_c l(\epsilon)/(l-1)$ of the schema offspring will have a crossover falling within its defining positions. When an instance of ϵ is crossed with another instance of ϵ the result is also an instance of ϵ, otherwise the result may or may not be an instance of ϵ. The probability of ϵ crossing with ϵ is just $P(\epsilon,t)$ so no more than a proportion $(1-P(\epsilon,t))P_c l(\epsilon)/(l-1)$ of the modified offspring of ϵ can be expected to be instances of schemata other than ϵ, the remainder, $[1-(1-P(\epsilon,t)P_c l(\epsilon)/(l-1)]$, will be instances of ϵ. Therefore, if N is the number of solutions in the population and $N'_\epsilon(t+1)$ is the number of instances of ϵ that survive into the next generation:

$$\begin{aligned} P(\epsilon,t+1) = \frac{N'_\epsilon(t+1)}{N} &\geq \left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right) \frac{N_\epsilon(t+1)}{N} \\ &\geq \left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right) \frac{\bar{\mu}_{\epsilon t} N_\epsilon(t)}{\bar{\mu}_t N} \\ &\geq \left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right) \frac{\bar{\mu}_{\epsilon t}}{\bar{\mu}_t} P(\epsilon,t). \end{aligned}$$

Of course, it is possible that some instances of ϵ could form from the crossover of two solutions that did not contain ϵ but this would only strengthen the inequality.

As Holland points out, instances of a schema will increase so long as:

$$\left(1 - \frac{[1 - P(\epsilon, t)]P_c l(\epsilon)}{(l-1)}\right) \frac{\bar{\mu}_{\epsilon t}}{\bar{\mu}_t} \geq 1.$$

Therefore, a schema will become more populous if:

$$\bar{\mu}_{\epsilon t} \geq \frac{\bar{\mu}_t}{\left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right)}.$$

The denominator of this expression is equal to $1 - c$ where c is the product of three probabilities and so $c \leq 1$. Also, $1/(1-c) \geq (1+c)$ since $(1-c)(1+c) = (1-c^2) \leq 1$ which gives:

$$\bar{\mu}_{\epsilon t} \geq \bar{\mu}_t \left(1 + \frac{[1 - P(\epsilon, t)]P_c l(\epsilon)}{(l-1)}\right).$$

Taking the worst case with $P_c = 1$ and assuming that $P(\epsilon, t)$ is small for any given ϵ since the number of schemata is in general very large, then to ensure increase it is required that:

$$\bar{\mu}_{\epsilon t} \geq \bar{\mu}_t \left(1 + \frac{l(\epsilon)}{(l-1)}\right)$$

This demonstrates the intrinsic parallelism of the genetic algorithm in that the proportions of each schema increase or decrease according to the above, independently of what is happening to all the other schemata in the population. Further, the relative proportions with which a schema appears in the population is dependent on its past performance and thus serves as a record of that performance.

The effect of mutation on this result is given by the following lemma.

Lemma 5.5.1 *(Holland 1975) For a reproductive plan of type R, using both the crossover and mutation operators, the expected change in $P(\epsilon,t)$ over one generation is bounded by:*

$$P(\epsilon,t+1) \geq \left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right) \frac{\bar{\mu}_{\epsilon_t}}{\bar{\mu}_t} P(\epsilon,t)(1-P_m)^L.$$

Where P_m is the probability of a mutation occurring at any given locus, and L is the number of defining positions for schema ϵ.

Proof – From the previous proof we know that the term,

$$\left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right),$$

is the probability of an instance of ϵ surviving crossover. The probability of one of the defining loci being mutated is P_m, so the probability of each defining position being unchanged is $(1-P_m)$, and the probability of all L loci being unchanged is $(1-P_m)^L$. Hence the proportion of instances of schema ϵ in a population after crossover and mutation is:

$$\left(1 - \frac{[1-P(\epsilon,t)]P_c l(\epsilon)}{(l-1)}\right)(1-P_m)^L.$$

Substituting for this term n theorem 5.5.1 gives the desired result.

Clearly mutation is a constant source of loss of schemata, and hence the reason for keeping the value of P_m small. However, some mutation is necessary to maintain diversity and lessen the chance of entrapment in false maxima.

5.6 Robustness

The ultimate level of robustness that could be required of a GA's reproductive plan is for it to converge to the optimal solution under any conditions. However, as Holland points out, this is a pointless performance measure since searches based on exhaustive evaluation will converge but are useless in practice. Further, Holland states that when data can be represented by no more than a population of N trial solutions, where N is less than the number of all possible solutions, then no plan can be guaranteed to yield convergence. Specifically, Holland gives the condition that for any $N < |\mathcal{K}|$ there exists $\delta(N) > 0$ such that:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} P(K^*, t) = 1 - \delta(M).$$

where K^* is a subset of $\mathcal{K}$ consisting of one or more structures with optimal performance which is to say structures $k^* \in \mathcal{A}$ such that the mean of $\mu(k^*)$ is at least as high as the mean for any $k \in \mathcal{K}$.

This occurs because for any finite number of trials of a sub-optimal solution there is a non-zero probability that its observed average performance will exceed that of the observed value for the optimal solution(s). There is thus a non-zero probability that enough sub-optimal solutions could be wrongly observed to lead to the deletion of data pertaining to the optimal solutions(s). Thus it cannot be expected for a genetic algorithm to produce optimal solutions even given an arbitrary large number of generations.

However, $\delta(N') < \delta(N)$ for $N' > N$ even when N' is less than the number of all possible solutions. This is because:

1. More copies of sub-optimal solutions in a population enables the average performance to be calculated with more accuracy. If the variance of $\mu(k)$ is smaller, then there is less chance that it will be observed that $\bar{\mu}(k) > \bar{\mu}(k^*)$.

2. The bigger the population the more generations that are required to completely displace optimal solutions which means that $\bar{\mu}(k)$ has to exceed $\bar{\mu}(k^*)$ over a longer period which is less likely.

To summarise the results of the last few sections, during each generation a GA evaluates and updates a large number of schemata in an intrinsically parallel way, the current generation acting both as the set of best solutions found to date as well a history of the search, represented by the relative proportions in which each schema appears. Whilst this makes for an algorithm capable of quickly finding good solutions, it will not in general find optimal ones in domains with an infinite or unmanageably large number of possible solutions due to practical constraints on population size.

5.7 Limits on Implementations

Holland identifies some of the factors that can adversely affect the performance of a GA as follows:

- $\mathcal{K}$ is large resulting in many alternatives to test.
- The solutions $k \in \mathcal{K}$ are complicated so that it is difficult to determine which schema or components are responsible for good performance.
- $\mu(k)$ is a complicated function containing many interdependent parameters and hence can be non-linear and discontinuous.
- $\mu(k)$ can be a time and space varying function.
- E presents the adaptive plan, τ, with a great flux of information.

These problems are however purely theoretical in that they identify limitations on a GA's performance due to the search domain, fitness function, and reproductive plan, but do not consider the practical limitations imposed by an

implementation. Booker (Booker 1987) has addressed such problems and identifies the main one to be that of the population converging prematurely to solutions that are far from optimal. This is due to a combination of the effect described in the last section in which inferior solutions can be observed to out-perform superior ones, and the fact that any practical implementation of a GA must have a finite population size and hence cannot allocate an arbitrarily small number of offspring to each solution. For example, only an integer number of offspring can be generated, and for a population of fixed size, this means that some solutions must necessarily get allocated zero offspring. This complete loss of solutions never occurs in the theoretical formulation, the proportions of inferior solutions in the population simply become very small but still significant should one of their number eventually be transformed into a much fitter solution. The divergence of the search trajectory from that predicted and the domination of far from optimal solutions is termed genetic drift, after its analogue in population genetics.

The problem is aggravated if, during the early generations, a solution should appear that is significantly better than the average although not particularly good in absolute terms. Due to the way offspring are allocated, such a solution contributes a large proportion of offspring to the next generation which in turn are likely to be of above average fitness. After a few generations the descendants of an abnormally good solution have completely dominated the population by pushing out all other solutions. The resulting lack of diversity ensures that the population will converge prematurely. The fraction of the next generation which are descended from a single solution is defined by Baker (Baker 1987) as the 'percent involvement' and can be used as an early warning of premature convergence.

Several approaches to alleviating the problem are discussed by Booker, the simplest of which is to simply increase the population size to allow for the representation of smaller shares of the space available. This is limited by efficiency considerations since larger populations require more memory and take longer to evaluate. In addition this detracts from the appeal of a GA - that of achieving near optimal solutions without recourse to mammoth evaluation. Initially, increasing the mutation rate would appear to prevent premature convergence by increasing the amount of disruption in the system, however, mutation affects good and bad solutions equally and as verified in the next chapter increasing its probability above a background level has an overall detrimental effect on the performance of the algorithm. Booker briefly mentions other techniques for delaying convergence but they involve such ideas as introducing extra rules to the adaptive plan that weigh in favour of exploration over exploitation. Bookers preferred approach is to improve performance by careful implementation of the crossover operator. The three techniques suggested are the use of two-point crossover (DeJong 1975), variable crossover rates, and the maximising of change during crossover operations.

Two-point crossover is a slightly modified version of the operator described by Holland, in that both crossover endpoints are chosen at random. The implementation of two-point crossover is explained in more detail in the next chapter. The crossover rate is the frequency with which the operator is applied and by varying this inversely with changes in the percent involvement Booker is able to reduce the influence of well above average solutions and thus lessen the risk of premature convergence. Bookers final modification of crossover involves ensuring change from each application of the operator. For example, when the crossover sections of two solutions are the same no new schemata are generated for trial

and the hence the process is ineffectual. By deriving 'reduced surrogates' which is to say the sections of each solution containing non-matching alleles, and forcing the choice of crossover endpoints to lie in these sections, Booker ensures that the crossover operator always produces new schemata for trial. Of these three techniques we will make use of two-point crossover since Booker reports a significant improvement in best-so-far performance at very little extra computational expense. However, ensuring change on crossover is reported as having little or no effect on best-so-far solutions, and varying the crossover rate adds computational expense to what proves to be an already slow program and so neither of these techniques will be used.

One aspect of GA implementation that appears to be have been overlooked is that of the exact numbers of offspring to be assigned to each solution. Theory suggests that the numbers should be proportional to the fitness of each solution but what should the proportionality factor be? The question is important since the number of offspring allocated to the best solution will determine the fitness level at which solutions start to get zero offspring and as the foregoing discussion indicates this can lead to loss of diversity, genetic drift and subsequent premature convergence.

In the next section we propose a reproductive plan for a GA involving no arbitrary choices in the allocation of offspring and which attempts to keep to a minimum the advantage given to above average solutions.

5.8 An Alternative Plan: D_1

The problem with offspring allocation in a population of fixed size would appear to be that as soon as one solution is given more than a single offspring

another solution must necessarily get zero. However, we propose that this problem is illusory and caused by the focusing of attention on the wrong quantity – that of the number of offspring – and that the correct factor to consider is the total amount of genetic material each solution imparts to the next generation. This is justified when we realise that after the use of a crossover operator the resulting child solution should be counted as an offspring for both parents.

From the definition of crossover it can be seen that the resulting child contains genetic material from both parents to an extent depending on the selected crossover length. Since the crossover length is permitted to be anything between zero and the full length of a solution, it is impossible to tell from an inspection of the child and its parents exactly which of the parents was selected due to its superior fitness, and which simply as a randomly selected mate. Hence the only consistent way of classifying the child is as the offspring of both parents and not simply the above averagely fit one.

If we then consider each parent as having one of two shares in each child it is involved in producing, then we have the child generation containing a total of $2N$ shares where N is the population size. Clearly now every parent solution is able to be allocated at least one share in the next generation whilst there being adequate capacity for allocating extra shares on the basis of relative fitness. Further, this allocation can be achieved in a natural way by the use of the following reproductive plan which we will call D_1.

Reproductive Plan $\mathbf{D_1}$

1. Set $t = 0$ and initialise the population randomly.
2. For each solution, $k_n(t)$, select at random a set of r solutions, $K_r(t)$, from the current population, $K(t)$, where $r > 1$.
3. Select the fittest solution, $k^*(t) \in K_r(t)$.
4. Apply the crossover operator to $k_n(t)$ and $k^*(t)$ and store the result in the next generation as $k_n(t+1)$.
5. Apply the mutation operator to $k_n(t+1)$ with probability P_m.
6. Increment t by one.
7. Repeat steps two to six until $t = T$ where T is the number of generations to be run.

To analyze the performance of this plan we need the following definition.

Definition 5.8.1 Let the number of solutions in a population at time t with fitness values between μ and $\mu + \delta\mu$ be $\mathcal{N}_t(\mu)$. Then define:

$$\nu_{nt} = \frac{1}{N} \int_0^{\mu_{nt}} \mathcal{N}_t(\mu) d\mu;$$

where N is the population size and μ_{nt} has its usual meaning. (We have assumed here that the μ_{nt} are positive, real numbers, but in general the lower limit of the integration is the lower limit of the range of fitness values.)

From its definition ν_{nt} is clearly the probability that solution $k_n(t)$ is fitter than any other single solution chosen at random from the current population. We now state the following theorem concerning the number of shares each solution can expect to get in the next generation.

Theorem 5.8.1 *For a reproductive plan of type D_1, the expected number of population shares allocated to each solution is bounded by:*

$$N(k_n(t), r) \geq 1 + (N^r + (N-1)^r)\left(\frac{\nu_{nt}}{N}\right)^{r-1}.$$

where N is the population size.

Proof – The probability of a given solution $k_n(t)$ not being selected at all in a random sample of r solutions from a population of size N is $(N-1)^r/N^r$, and so the probability of it being chosen at least once is $1-(N-1)^r/N^r$. At worst there can be only $r-1$ other distinct solutions in the random sample and so the probability of $k_n(t)$ being the fittest is better than ${\nu_{nt}}^{r-1}$. Therefore the number of shares allocated to $k_n(t)$ by random selection over N trials is greater than or equal to:

$$N\left(1-\frac{(N-1)^r}{N^r}\right)\nu_{nt}^{r-1} = (N^r + (N-1)^r)\left(\frac{\nu_{nt}}{N}\right)^{r-1}.$$

Finally, each solution, $k_n(t)$, is guaranteed one share in $k_n(t+1)$ and so its total number of expected shares is:

$$N(k_n(t), r) \geq 1 + (N^r + (N-1)^r)\left(\frac{\nu_{nt}}{N}\right)^{r-1}.$$

For brevity we shall henceforth write $c(r) = N^{1-r}(N^r - (N-1)^r)$ which gives the number of population shares for each solution as $1 + c(r)\nu_{nt}^{r-1}$

Lemma 5.8.1 *The condition necessary for a schema to increase its probability from one generation to the next under reproductive plan D_1 is:*

$$c(r)\langle\nu_{\epsilon t}^{r-1}\rangle \geq \left(1 + \frac{l(\epsilon)}{l-1}\right);$$

where $\langle \nu_{\epsilon t}^{r-1} \rangle$ *is the average value of* ν_{nt}^{r-1} *for all* $k_n(t)$ *that are instances of* ϵ.

Proof – The effect of plan D_1 can be interpreted in Holland's terms as allocating at least $c(r)\nu_{nt}^{r-1}$ offspring to each solution, where the mate is chosen deterministically rather than at random, thus ensuring that all solutions get a share in the next generation. Following the same procedure as in theorem 5.5.1, the expected number of offspring for the set of solutions containing schema ϵ is:

$$N_\epsilon(t+1) \geq \sum_{n:k_n(t)\in B_\epsilon(t)} c(r)\nu_{nt}^{r-1} = N_\epsilon(t)c(r)\langle \nu_{\epsilon t}^{r-1} \rangle.$$

Even though the mate for each solution is chosen deterministically, the probability that it will be an instance of ϵ is still $P(\epsilon, t)$, and since the crossover and mutation operators of D_1 are the same as those for R_1 and R_2, the probability that a given crossover will contain new instances of ϵ is the same as in theorem 5.5.1 but with $P_c = 1$. Hence:

$$P(\epsilon, t+1) \geq \left(1 - \frac{[1 - P(\epsilon,t)]l(\epsilon)}{(l-1)}\right) c(r)\langle \nu_{\epsilon t}^{r-1} \rangle P(\epsilon,t)(1-P_m)^L.$$

For increase we need the factors on the right hand side to be greater than one. Once again re-expressing the terms in brackets and assuming $P(\epsilon, t)$ is small for any given ϵ we obtain:

$$c(r)\langle \nu_{\epsilon t}^{r-1} \rangle \geq \left(1 + \frac{l(\epsilon)}{l-1}\right)(1-P_m)^{-L}.$$

However, the mutation probability is usually very low and so by assuming that $(1-P_m)^{-L} \approx 1$ we obtain the required result.

The advantages of reproductive plan D_1 are as follows:

1. Each solution gets a share in at least one solution in the next generation regardless of how low its fitness value may be. This guarantees that no solution gets completely neglected during a single generation.

2. Each solution gets a total number of shares in the next generation dependent on its relative fitness ensuring that better solutions still donate more genetic material than do poorer ones.

3. There is no need to make an arbitrary choice of the exact numbers of offspring to allocate to each solution since this is handled automatically.

4. By choosing r to be small the advantage of good solutions over poor ones can be reduced so encouraging exploration over exploitation and lessening the risk of premature convergence. For example, with $r = 2$ the expected number of shares for solution $k_n(t)$ is:

$$1 + \frac{(2N-1)\nu_{nt}}{N} \approx 1 + 2\nu_{nt};$$

if N is large. Clearly no matter how much better a particular solution is than the rest it can have $\nu_{nt} = 1$ at best and so contribute on average only three shares to the next generation. A reproductive plan in which offspring numbers are allocated proportional to $\mu_{nt}/\bar{\mu}_t$ could have the best structure contributing to nearly all of the offspring.

5. Since only the fitnesses of the r solutions in the randomly chosen set need be known at any one time, it is only necessary to evaluate a solution if it is chosen as part of one of these sets. Hence, a solution that never gets selected need not be evaluated. For example, consider again the case when $r = 2$. The probability of selection for each structure is $1/N$ each time. In total there are $2N$ selections and so the probability of a structure not being picked at all during one generation is $(1 - 1/N)^{2N}$. Typically N is of the order 100 and gives a 0.134 probability of a

solution not being evaluated. For complex fitness functions their evaluation is the major time consuming component of a GA and so any reduction in the number of evaluations that are necessary will result in improved run times.

6. Since it is the best solution from the random subset of the whole population that is required, it is not necessary to completely evaluated the fitness of each solution but only to be able to determine which is the better of two given solutions permitting the use of approximate evaluation techniques.

Of course, there are also certain disadvantages associated with a reproductive plan of type D_1, namely:

1. The use of very small sample sizes makes it possible that a good solution could be allocated fewer shares than it warrants or that it could be allocated just its single share. However, this is the price we pay to reduce the effects of over exploitation.

2. Since not all trial solutions get evaluated the best-so-far performance will be degraded, especially for programs employing large populations and a small number of generations.

5.9 Summary

In this chapter we have presented a summary of Holland's work on the theory of genetic algorithms and introduced the characteristics that make them suitable for the intended application to collage construction. Namely, the ability to conduct efficient searches resulting in the attainment of near optimal solutions in large and complex domains for which no guiding theory is available, but in which trial solutions are readily evaluated.

The terminology of genetic algorithms has been introduced and their power explained by the way in which they allocate exponentially increasing numbers of trials to above average solutions in an intrinsically parallel fashion.

We have further presented a discussion of the problems encountered with practical implementations such as genetic drift and premature convergence, and have suggested our own reproductive plan, D_1, in an attempt at improving program performance and efficiency.

The next chapter concerns the implementation of a genetic algorithm incorporating a reproductive plan of type D_1, and with the purpose of constructing full shape collages.

6 IFS ENCODING BY GENETIC ALGORITHM

In this chapter we describe the implementation and testing of a GA designed to automatically generate IFS representations of general shape input, and which incorporates a reproductive plan of type D_1. The main aim of the implementation is to assess the practicality of the encoding method rather than an investigation of the properties of genetic algorithms or reproductive plans in themselves. However, some of the issues discussed include the determination of practical program parameters, the trade-off between the accuracy of fitness evaluation and population size, and the effect of different fitness functions on best-so-far program performance. A notated listing of the C source code for the program can be found in appendix A.

6.1 Program Parameters

The search domain for a GA intended to solve the inverse problem is a space in which the points correspond to sets of contraction mappings. Regardless of practical considerations, the purely theoretical limitations of GAs discussed in the last chapter indicate that it would be naïve to expect anything like optimal solutions from such a search. However, we can simplify the problem, and hopefully increase the chance of obtaining near optimal solutions, by fixing the number of mappings to be used, which is to say fixing the 'size' of the collages. We can justify this by referring to the discussion of chapter three in which it was

stated that, in general, increasing the number of mappings in a collage increases the resolution of the representation and it was suggested that the resolution to which an encoding program is required to work should be the determining factor in deciding collage size. Thus we would normally expect an encoding program to search for the best collage possible with a fixed number of mappings.

We therefore take collage size as one of six program parameters to be determined at runtime, the full set being:

POP - population size.

GEN - number of generations to run.

MAP - collage size.

XLN - crossover length.

MUT - mutation probability.

SUB - subsampling factor.

Apart from *MAP*, *SUB* is the only parameter whose use is not obvious from the discussion of the previous chapter. Its value is a measure of the accuracy to which the fitness of each solution is evaluated and will be discussed in detail later.

Clearly trying to optimise six independent variables is a difficult task in itself, complicated in this case by the stochastic nature of a GA and the difficulty of assessing the performance of each parameter combination. As a simplification we impose constraints on the amount of computer resource that we are prepared to spend on the encoding of each shape, the practical result of which is that we place a limit on program runtimes. By doing this we necessarily reduce the maximum performance that it may be possible to extract from the program, but in exchange we have a useful framework within which to make comparisons.

Further, there is little point in achieving good results if an exorbitant amount of resources are required.

In terms of the given parameters, and for a fixed value of MAP, we have approximately:

$$t \propto (GEN \times POP);$$

where t is the program runtime. The proportionality is only approximate since the other parameters, (especially SUB), complicate the relation as discussed more fully in a later section. However, what we can achieve by fixing the value of $(POP \times GEN)$ is a baseline runtime from which we expect small variations as the other parameters are changed. Specifically, we take $POP = GEN = 100$ as our standard values. Some justification for this is provided by Fitzpatrick and Grefenstette (Fitzpatrick and Grefenstette 1988) who report that these are typical values and ones for which useful results can be expected, since they allow the evaluation of ten thousand trial solutions. With MAP, POP, and GEN fixed there are now only three other parameters that need to be set which constitutes a much simpler problem. The determination of values for MUT, XLN, and SUB is covered in sections 6.6 and 6.7, but for the present we proceed by describing the implementation of the GA components given in the last chapter in terms of the above parameters.

6.2 Program Environment

The environment of a GA is the source of information on which the fitness of trial solutions is evaluated and hence it determines the future evolution of the initial population. In theory therefore, we must associate it with a specific shape input which is to say, $E = S \in \mathcal{H}(\mathbf{R}^2)$ and hence, $\mathcal{E} = \mathcal{H}(\mathbf{R}^2)$, since we make

no restrictions as to the type of shapes on which the program can operate. In practice though, we take the environment to be an image file created by a simple image processing program and which contains the following information:

1. The number of pixels that constitute the shape.
2. The coordinates of the centroid of the shape in the image plane.
3. The coordinates of each pixel relative to the centroid.
4. The maximum extent of the shape in the x- and y-directions.

The information in the image file in addition to the minimum requirement of the pixel positions is a necessary consequence of the conventions adopted in chapter four, and which we have retained for the current implementation. The value of all coordinates in the image file are scaled by a factor of 1024 to allow the use of integer arithmetic thus increasing execution speed without loss of accuracy. Since the environment is time independent, the information in the image file is invariant and need only be read once at the beginning of a program run.

6.3 Solution Representation

Following convention we use a binary representation for the solution strings, and as a result of the discussion of chapter four we continue to use affine transformations. We are able to keep the representation scheme simple because an IFS is basically a list of mapping coefficients which can be translated directly into a binary string. Since two-dimensional affine transformations requires six numbers to be completely specified, solution strings consist of $(6 \times MAP)$ sections each containing a fixed number of bits which represent one mapping coefficient. The number of bits used is important since it determines the resolution to which

mapping coefficients can be calculated and hence limits the accuracy to which the program works.

The work of Barnsley (Barnsley 1988) indicates that a coefficient accuracy of two decimal places is sufficient to produce accurate collages, and this is supported by the demonstration of code robustness in chapter three which shows that changes made in the second decimal place have little effect on the attractor as a whole. Using the C programming language on our system, the 'char' data type is of length eight bits and consequently takes signed values in the range $[-128, 127]$. This makes it convenient for use since it gives a coefficient resolution of $1/128 \approx 0.008$. Using one eight bit byte for each mapping parameter results in the length of each solution string being $48 \times MAP$ bits.

For the choice of coordinates and metric made, the range of values that each mapping parameter may take are as follows:

$$r_1, r_2 \in (-0.707, 0.707) \qquad \theta_1, \theta_2 \in [-\pi, \pi],$$

$$x_0 \in [xmin, xmax] \qquad y_0 \in [ymin, ymax].$$

where xmin, xmax, ymin, ymax, are the shapes extent as given in the image file. The restricted range for the scale factors, r_1 and r_2, ensures that any legal combination of parameters results in a strictly contractive mapping.

The solution strings are decoded in the following way. If the value of the nth byte of a group of six is denoted by c_n, then the values of r_1 and r_2 are given by:

$$r_1 = \frac{c_1}{128.0\sqrt{2}} \qquad r_2 = \frac{c_2}{128.0\sqrt{2}}.$$

The next two bytes are taken as the angle parameters, θ_1 and θ_2 and their values are obtained from the following:

$$\theta_1 = \frac{c_3 \times \pi}{128.0} \qquad \theta_2 = \frac{c_4 \times \pi}{128.0}.$$

Of course, the fact that a value of $+\pi$ can never be attained is unimportant since a value of $-\pi$ produces exactly the same effect. The final two bytes of a group represent the translational components of the mapping and are decoded as follows:

$$x_0 = \frac{(c + 128.0)}{255.0} * (xmax - xmin) + xmin;$$

$$y_0 = \frac{(c + 128.0)}{255.0} * (ymax - ymin) + ymin;$$

which gives them their proper full range. Clearly such a solution representation scheme satisfies the condition of being one-to-one and onto, in that every legal string represents a contractive collage, and every possible strictly contractive collage defined to two decimal places of accuracy has a representation by a string.

6.4 Implementation of D_1

Theorem 5.8.1 tells us the number of shares in the next generation that can be expected for each solution under a reproductive plan of type D_1 given a random sample size or r. If we consider these numbers at the extremes of the permitted range of r we find the following:

$$N(k_n(t), 2) = 1 + \frac{(2N-1)}{N}\nu_{nt} \approx 1 + 2\nu_{nt} \qquad \text{(assuming large } N\text{)};$$

and

$$\lim_{r \to \infty} N(k_n(t), r) = N + 1 \qquad \text{if } \nu_{nt} = 1.0$$

$$= 1 \qquad \text{if } \nu_{nt} < 1.0.$$

This clearly demonstrates that as the sample size is increased the best solution gets a greater proportion of the 'quota' shares until in the limit it is allocated them all and the other solutions only get their one guaranteed share. (Notice that in both extremes the total number of shares is still equal to $2N$). The limit case corresponds to the fittest solution always being chosen as the mate and will

clearly result in that solutions descendants dominating the population. With $r = 2$ however, the average number of extra shares the best solution can hope to get is two, hence restricting its ability to become dominant too quickly. This is just the effect required to prevent premature convergence and it is therefore proposed to adopt the use of a sample size of $r = 2$ henceforth. By combining such a small sample size with the ability to use approximate evaluation, and only evaluating solutions as and when necessary, the implementation of plan D_1 is made highly efficient.

The decision to take $r = 2$ is further supported by noticing that for large N the result of lemma 5.8.1 now simplifies to the condition:

$$2\bar{\nu}_{\epsilon t} \geq 1 + \frac{l(\epsilon)}{l-1};$$

where $\bar{\nu}_{\epsilon t}$ is the average fitness of all the instances of schema ϵ at time t. The interpretation of this is that for even the shortest schemata to increase their population share, it is necessary that, on average, solutions containing their instances be more likely to be chosen in preference to any other randomly selected solution. It should be noted at this point that although setting $r = 2$ gives a theoretical minimum advantage to above average solutions, it is possible that this goes to far and results in adverse effects on the algorithms performance. An extensive investigation of the effects of sample size is however beyond the scope of the current work, and we rely instead of the satisfactory algorithm performance described later as an indication that $r = 2$ is not too small.

With the size of r set, we now describe the implementation of the crossover and mutation operators. As mentioned in chapter five, we do not use the simple crossover algorithm described by Holland, but a slight variant introduced by DeJong (DeJong 1975) called two-point crossover, the difference being that now

both the start and end points of a crossover section are determined at random. The rationale for this algorithm becomes apparent if the solution strings are considered to be joined at their ends to form loops. A crossover section can now be any segment of a loop and is not restricted to the arbitrary choice of one bit being the fixed endpoint. This different crossover technique does not change the result of theorem 5.4.1 since we simply associate positions inside the two crossover points with those to the right of the single crossover point, and points outside with those to the left, or vice-versa, and the mathematics remains the same. However, two-point crossover differs in effect to the standard algorithm since it has a less disruptive effect on long schemata. For example, with a fixed crossover endpoint, even for short crossover lengths, any schemata with a defining position at this endpoint is nearly always disrupted. By varying the position of the crossover endpoint this effect is distributed more evenly amongst all the schemata in the population and results in improved performance (Booker 1987).

Using the string representation scheme given in the previous section, the simplest crossover implementation would be one in which two bytes along a string were chosen at random and the intervening bytes swapped between two solutions. However, this only allows crossovers in lengths of multiples of eight bits and would rather circumvent the use of a binary representation. To ensure crossovers of any length, masks are created for the end bytes of a crossover section. Thus the algorithm used to implement two-point crossover is:

1. Select a random integer, x_1, between one and $6 \times MAP$.

2. Select a second random integer, x_2, between zero and XLN.

3. Starting at the x_1th byte, crossover x_2 bytes between parents.

4. Place the child solution in the next generation.

5. Generate a random left-end mask byte.

6 Apply to the byte to the left of the childs crossover section.

7. Generate a random right-end mask byte.

8. Apply to the byte to the right of the childs crossover section.

We choose to determine the endpoint of the crossover in terms of a distance from the start point, since then the parameter XLN controls the amount of genetic information that is inserted into each deterministically selected solution and so will affect the speed with which good solutions become dominant. The range of permitted values for XLN is between zero and $6 \times MAP - 2$. At the lower limit the amount of crossover is determined by the end masks alone and so the total crossover length cannot be greater than two bytes. When XLN has its maximum value, it is possible for nearly the whole length of two solutions to undergo crossover.

As an example of the operation of the end masks, let the bytes immediately to the left of the crossover section for two solutions be c and c' respectively. If the bit values in these bytes are represented by $a_1, a_2, \ldots, a_8$ and $a'_1, a'_2, \ldots, a'_8$, and the mask byte, m, is 00000111, then the operation:

$$(c \wedge m) + (c' \wedge (\neg m));$$

results in a byte with the structure:

$$a'_1, a'_2, \ldots, a'_5, a_6, a_7, a_8;$$

hence adding three more bits to the crossover length to the left. The left-end masks are generated by choosing a random number, $x \in \{0, 1, \ldots, 7\}$, and setting their value equal to $2^x - 1$. The same operation handles the right most byte of

the crossover using masks consisting of the binary values of -2^x for x chosen at random as before. With the endpoint crossover defined in this way, the minimum number of bits that are exchanged is one, corresponding to the left-end mask having a value of zero, and the right-end mask being -128. The maximum number of bits that the end mask can exchange is fifteen when the values are 127 for the left-end, and -1 for the right-end. Thus the minimum and maximum total lengths of a crossover section are one bit, and the length of the string minus one bit respectively.

Mutation is handled in a similar way to crossover by the use of masks. Each child solution is given a probability of MUT of undergoing mutation. If mutation is selected then the following algorithm is employed:

1. Select a random integer, x, between one and $6 \times MAP$.

2. Generate a random mask.

3. Apply the mask to xth byte of the child solution.

A mutation mask is generated by setting its value to 2^y where y is chosen at random from $\{0, 1, \ldots, 7\}$. If c is the byte selected to undergo mutation and 00100000 is the mask then the operation:

$$(c \wedge (\neg m)) + ((\neg c) \wedge m);$$

would change a byte 01101101 to 01001101. This is a slightly different form of mutation to that described in the last chapter since it ensures that at most one bit of a solution is changed, however this is not of great significance since the very low mutation rates usually employed make multiple mutations of a single solution an extremely rare occurrence. The chance of any given bit of a solution being mutated is $MUT/(48 \times MAP)$ which is the value that is intended by the factor

P_m in lemma 5.5.1 so and the value of MUT will typically be much larger than is normal for a mutation parameter. The advantage of implementing mutation in this way is that only two calculations need to be made per solution rather then the $48 \times MAP$ needed if each bit were to be selected for mutation individually.

6.5 Fitness Functions

The obvious choice of fitness function for testing trial solutions is that provided by the collage theorem, and so we could write:

$$\mu_{nt} = \frac{h(\mathbf{S}, W_{nt}(\mathbf{S}))}{(1 - s_{nt})}.$$

where $\mathbf{S}$ is the input shape and s_{nt} and W_{nt} are the minimum contractivity factor and the union of mappings for solution $k_n(t)$ respectively. However, the value of this function tends to infinity as the contractivity factor approaches unity which is fine theoretically, but unacceptable in practice. Also, although a minor point, the function decreases as the solutions improve and so it would require the minimisation of average population fitness. Taking the reciprocal of the function ensures that fitness needs to be maximised but now its value tends to infinity as the Hausdorff distance tends to zero. We therefore define the first of the fitness functions we consider to be:

$$\mathcal{A}_{nt} = \frac{(1 - s_{nt})}{1 + h(\mathbf{S}, W_{nt}(\mathbf{S}))}.$$

Adding one to the denominator dramatically changes the behaviour of the function when $h < 1$, but this is of no importance since by working in the discrete space of the pixel plane, unity is the smallest non-zero value that h can take. The advantage of this modification is that fitness values are always finite, and moreover are normalised to be in the range $[0, 1]$.

Using $\mathcal{A}_{nt}$ the fitness of a solution is calculated as follows:

1. Decode the solution string, $k_n(t)$, and calculate the mapping coefficients.
2. Calculate the contractivity factor s_{nti} for each mapping,

   w_{nti}, where $i = 1, 2, \ldots, MAP$.
3. Calculate the contractivity factor for the collage,

   $s_{nt} = \max\{s_{nti} : i = 1, 2, \ldots, MAP\}$.
4. Calculate the set of distinct points produced by the mapping $W_{nt}(\mathbf{S})$.
5. Calculate the Hausdorff distance, $h(\mathbf{S}, W_{nt}(\mathbf{S}))$.
6. Evaluate $\mathcal{A}_{nt}$ as defined above.

Referring back to the definition of the Hausdorff metric in chapter two, it can be seen that the main computational burden is the calculation of the minimum distance between a point and a set, and the obvious algorithm to achieve this, that of exhaustive evaluation, is $O(m)$ where m is the number of points in the set. For a shape consisting of a finite number of points and with a collage of MAP mappings, there are a maximum possible number of $m \times MAP$ points in the mapping of the shape under the collage. Hence, the evaluation of $h(\mathbf{S}, W_{nt}(\mathbf{S}))$ requires the calculation of $2m^2 \times MAP$ distances and is thus $O(m^2)$. Since even very small shapes can contain many thousands of points we require a more efficient method of calculating Hausdorff distances or else a good approximation technique.

Shamos and Hoey (Shamos and Hoey 1975) state that if no preprocessing is permitted then $O(m)$ is both upper and lower bound for the calculation of the distance from a point to a set. That is, we will not be able to find an algorithm that will give an exact evaluation of $h(\mathbf{S}, W_{nt}(\mathbf{S}))$ that is better than $O(m^2)$. As for an approximation technique, it is known that the Hausdorff distance between two sets is just the distance between a pair of points, one in each set. This sensitive

dependence on just two points, and the fact that we make no restrictions on the type of shapes to be used as input, means that no useful approximations can be made and we must accept that evaluation of function $\mathcal{A}$ will be slow.

Although the Hausdorff metric appears in the theory of iterated function systems, we are not bound to its use in a GA. We therefore define the second of our fitness measures as:

$$\mathcal{B}_{nt} = (1 - s_{nt})\frac{m_{nt}}{m}.$$

where m_{nt} is the number of distinct points of $\mathbf{S}$ generated by $W_{nt}(\mathbf{S})$. Using $\mathcal{B}$ the fitness of a solution is found by the following algorithm:

1. Decode the solution string, $k_n(t)$, and calculate the mapping coefficients.
2. Calculate the contractivity factor s_{nti} for each mapping,
   w_{nti}, where $i = 1, 2, \ldots, MAP$.
3. Calculate the contractivity factor for the collage,
   $s_{nt} = \max\{s_{nti} : i = 1, 2, \ldots, MAP\}$.
4. Determine the number of distinct points in $\mathbf{S}$ that are
   produced by the mapping $W_{nt}(\mathbf{S})$.
5. Evaluate $\mathcal{B}_{nt}$.

Clearly $\mathcal{B}$ only performs a maximum of $m \times MAP$ calculations and is thus $O(m)$, which gives it a significant speed advantage over $\mathcal{A}$.

The final fitness function we will investigate is defined as:

$$\mathcal{C}_{nt} = \frac{m_{nt}(1 - s_{nt})}{m(1 + h(\mathbf{S}, W_{nt}(\mathbf{S}))};$$

which is just a hybrid form of $\mathcal{A}_{nt}$ and $\mathcal{B}_{nt}$. The performance of each of these fitness functions is discussed in section 6.8.

6.6 Parameter Settings

We now focus on the experimental determination of suitable values for the MUT and XLN parameters. We take $POP = GEN = 100$ and use the Sierpinski triangle as the test shape which sets the value of MAP at three. In order to first investigate the effects of varying MUT we set $SUB = 1$ and XLN at $6 \times MAP - 2$ which allows for exact evaluation and maximum crossover. Fitzpatrick and Grefenstette (Fitzpatrick and Grefenstette 1988) use the average fitness of the best fifty solutions in a population after a fixed number of generations as a performance measure for a GA, using this rather than the average fitness of the whole population to equalise the evaluation time for populations of different sizes. However, since we vary population size little, we measure program performance as the average fitness of all the solutions. Also, because we are not concerned at present with absolute performance, we use the cheapest of our fitness functions, $\mathcal{B}$, but with a slight modification. Since we are using a shape with a known IFS we can calculate the fitness of the optimal collage - which is clearly less than unity and dependent on the contractivity factor. By dividing through by this value we can better see the changes in performance.

The results of varying the mutation probability through the range zero to one is demonstrated in graphs C.1–C.5 in appendix C. We take the graph for $MUT = 0$ as our reference point since this shows the effect of having no disruption in the system. As expected the average fitness shows an initial smooth increase with an eventual leveling off at around the eighty-ninth generation to a value of approximately 56% of the optimal. An inspection of the attractors of the solutions in the final generation reveal almost complete convergence to a single highly sub-optimal solution and due to the lack of variation introduced into the system, there is no chance of any further improvement.

Introducing a mutation probability of 0.25, which is to say that on average 25% of the solutions in the population will have one bit changed each generation, produces the results of graph C.2. There is evidence of much greater fluctuation between the average fitness of successive generations but the final value is slightly higher than the previous result at around 65% of optimal. Graph C.3 shows the effect of setting $MUT = 0.5$ and now the deleterious effect of such a high mutation rate are clearly apparent, the fluctuations in average fitness are more pronounced, and the value after one hundred generations is approximately two-thirds of that achieve with $MUT = 0$. Inspection of the solutions in the final generation reveal large amounts of variation but clearly the tendency for improvement is becoming swamped by mutation effects. This trend continues with increasing values of MUT as depicted in graphs C.4 and C.5, and when finally $MUT = 1$ the population enters into an equilibrium between the opposing forces of refinement and disruption, with average fitness values fluctuating around 0.12. Graph C.6 shows the results obtained with a mutation probability of 0.01 which proved to give best results within the one hundred generation time limit.

Graphs C.7–C.12 show the effect of varying the value of XLN whilst keeping all other parameters fixed. With MAP set at three, the permitted range of XLN is integer values between zero and sixteen. For $XLN = 0$ Graph C.7 exhibits similar behaviour to that obtained with a one hundred percent mutation rate, in that fitness values fluctuate from generation to generation, and there is little discernible trend towards a steady rate of increase. This is probably to be expected since the exchange of very short string sections is little different from a mutation operation because early on at least, there is little correlation between a solutions fitness and the short schemata of which it is an instance. For $XLN = 2$ as depicted in C.8 the fluctuations are more pronounced, but there is an underlying

increase in the fitness of successive populations. As XLN is increased the absolute values of population fitnesses rise, the graphs become smoother, and the rate of increase higher. Best results are clearly obtained when crossover is allowed up to the full length of each solution string thus enabling the exchange of significant amounts of genetic information between parents and child.

The results of this section indicate that within the bounds we have imposed the values $MUT = 0.01$ and $XLN = 6 \times MAP - 2$ give best results. However, the indication is that in general the effects of mutation and crossover length have a sensitive dependence on each other and the remaining parameters such as POP and GEN. A more extensive investigation of this relationship is beyond the scope of this work and so we adopt the above values for all future program runs and now turn attention to the possibility of trading-off the accuracy of fitness evaluation for population size.

6.7 Evaluation vs Population

Fitzpatrick and Grefenstette (Fitzpatrick and Grefenstette 1988) have examined the benefits of trading accuracy in the evaluation of the fitness of each solution for an increase in population size and have shown that this can result in improved program performance in some cases. They assume that the runtime of a program is limited and hence that by making a quicker, more approximate fitness evaluation a larger population can be accommodated. Ignoring the overheads, they give the runtime of a GA as:

$$t \approx (\alpha + c\beta)GN.$$

where G is the total number of generations to be run, and N is the population size. The constant α is dependent on the amount of computation required to

produce each generation due to the operation of crossover and mutation. $c\beta$ is the cost of evaluating the fitness of each solution, where c is the number of sample points taken during the evaluation. It is assumed that the function being evaluated depends on the sampling of some quantity and that the accuracy to which the evaluation is made depends on the size of this sample. In practice though, all that is necessary is that the fitness function can be approximated and that c is a factor which represents the degree of approximation, decreasing as the evaluation becomes less precise.

For our implementation the runtime equation is modified to become:

$$t \approx (\alpha + c\beta)MAP \times GEN \times POP.$$

the only significant change being the introduction of the MAP term which makes explicit the effect of increasing solution string length. The evaluation functions we use take as their data the pixels of the input shape, and so an approximate evaluation is achieved by taking only a fraction of the available data points. Therefore, rather than refer to the absolute number of samples for an evaluation we set $c = m/SUB$ where m is the number of pixels that constitute the input shape and so corresponds to the maximum number of samples possible, and SUB is the subsampling factor. Clearly for a given value of MAP, if SUB is increased then the values of POP and GEN can be increased either individually or jointly without affecting the overall runtime. In general however, increasing the number of generations produces little improvement due to population convergence and so it is preferred to increase the value of POP alone. Fitzpatrick and Grefenstette point out that the amount of population increase that can be accommodated depends on the ratio of $\alpha/c\beta$. For example, if α is negligible and hence $\alpha/c\beta$ is small, which is to say that the major burden on the program is the evaluation of fitness, then POP varies inversely with SUB and large increases in population

are possible. However, if $n\beta$ is small then POP is effectively fixed regardless of the value of SUB.

Graphs C.13–C.16 show the effects of increasing the subsampling factor whilst keeping the population size constant and using the values for mutation and crossover determined earlier. With $SUB = 1$ we obviously get the the same results as before with the quick rise to high averages fitness values with that of the final generation being around 70% of optimum. Taking alternate data points produces the result of graph C.14 which shows a flattening of the rate of fitness increase and a leveling off at a slightly lower value of approximately 60% of optimum. Taking every fifth point significantly reduces the final average population fitness to 30% after one hundred generations whilst the graph for a subsampling factor of ten shows a rapid converges to a very low fitness value in the region of 15%. These results, whilst demonstrating the effect of inaccurate evaluation do not in themselves determine whether subsampling is detrimental. Graph C.15 in particular still shows an increasing fitness tendency right to the last generation and it is possible that given a larger population size it could perform as well as for more accurate evaluations. The following timings were obtained for each of the runs just described:

subsampling factor	runtime (cpu seconds)
1	981
2	533
5	332
10	274

Table 6.1 *Runtimes for a GA demonstrating the effect of increasing the subsampling factor.*

Using these timings we are able to calculate approximate values for α and β to be:

$$\alpha = (5.22 \pm 0.44) \times 10^{-3} \qquad \beta = (3.55 \pm 0.12) \times 10^{-5}.$$

Even though β is very small we are typically working with numbers of points in the region of one thousand and so significant increases in population size are still possible. For each of the three subsampling factors used, the possible population increase was calculated and the results of using these larger populations can be seen in graphs C.17-C.20. A subsampling factor of two allows seventy-five extra solutions in the population, but the resulting fitness plot does not vary significantly for that obtained with the standard population size. A set of 310 trial solutions with $SUB = 5$ and 417 for $SUM = 10$ give only marginal improvements to the program performance over one hundred generations, although both show the averages fitness to be still rising at that point which might indicate that more generations would give still better results. However, increasing the number of generations would require a further trade-off of between population and evaluation accuracy in order to keep the total runtime fixed, which would reintroduce the complexities of trying to optimise several different parameters. Our results indicate that for the types of fitness function used trading accuracy for the number of evaluations does not produce any benefit, and in the case of large subsampling factors, significantly reduces performance even when population sizes are increased to the maximum allowable. Unless otherwise stated the following parameters are now set for each program run:

$$POP = 100 \quad GEN = 100 \quad MUT = 0.01 \quad XLN = 6 \times MAP - 2 \quad SUB = 1.$$

The value of MAP is of course determined by the size of the optimal collage for each input shape.

6.8 Program Performance

Having set suitable values for the program parameters we now examine the best-so-far performance for each of the fitness functions. We have used another set of test shapes consisting of the attractors of known iterated function systems. These codes can be found in appendix B together with pictures of their collages and attractors. The set includes some well known fractals such as the Sierpinski triangle and the 'twin dragon' as used by Barnsley (Barnsley 1986), but also contains six 'random' fractals which were generated by choosing arbitrary mapping coefficients. Although the exact codes for each of the test shapes are known, we do not use this information to normalise the fitness values to fractions of the optimum but instead show absolute values.

Since it is the quickest and simplest of the functions defined, we start with function $\mathcal{B}$ which keeps average runtimes down to approximately 1800 seconds. Its performance in encoding each of the test shapes is demonstrated by graphs C.21-C.31 which plot average population fitnesses, and by figures 6.1 and 6.2, which show the collages and attractors of the best-so-far solutions after one hundred generations.

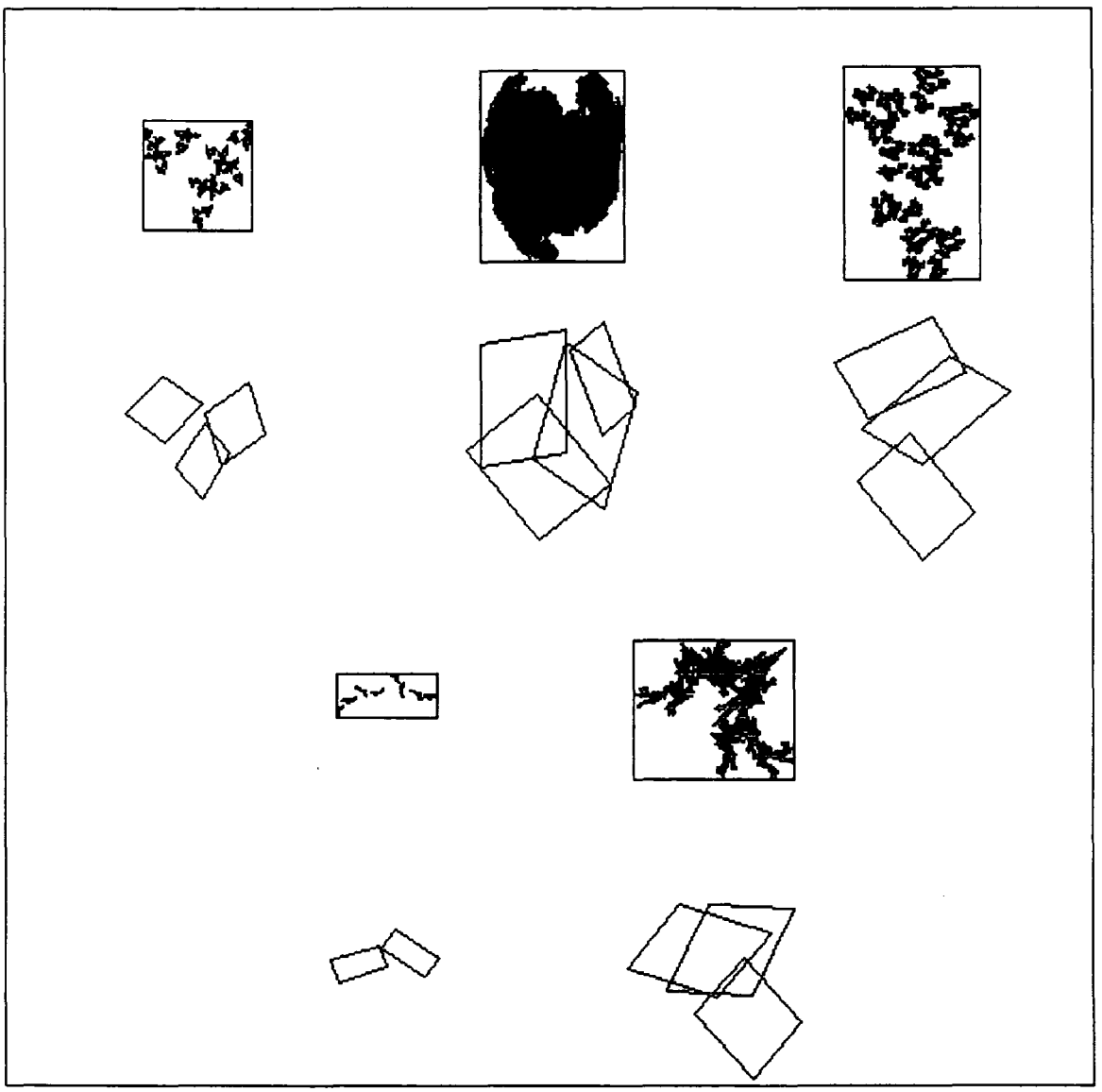


Figure 6.1 *The best-so-far solutions (attractors and collages) for encodings of the first five shapes of the test set using fitness function $\mathcal{B}$ The layout is the same as that for figure B.1 so a direct comparison can be made.*

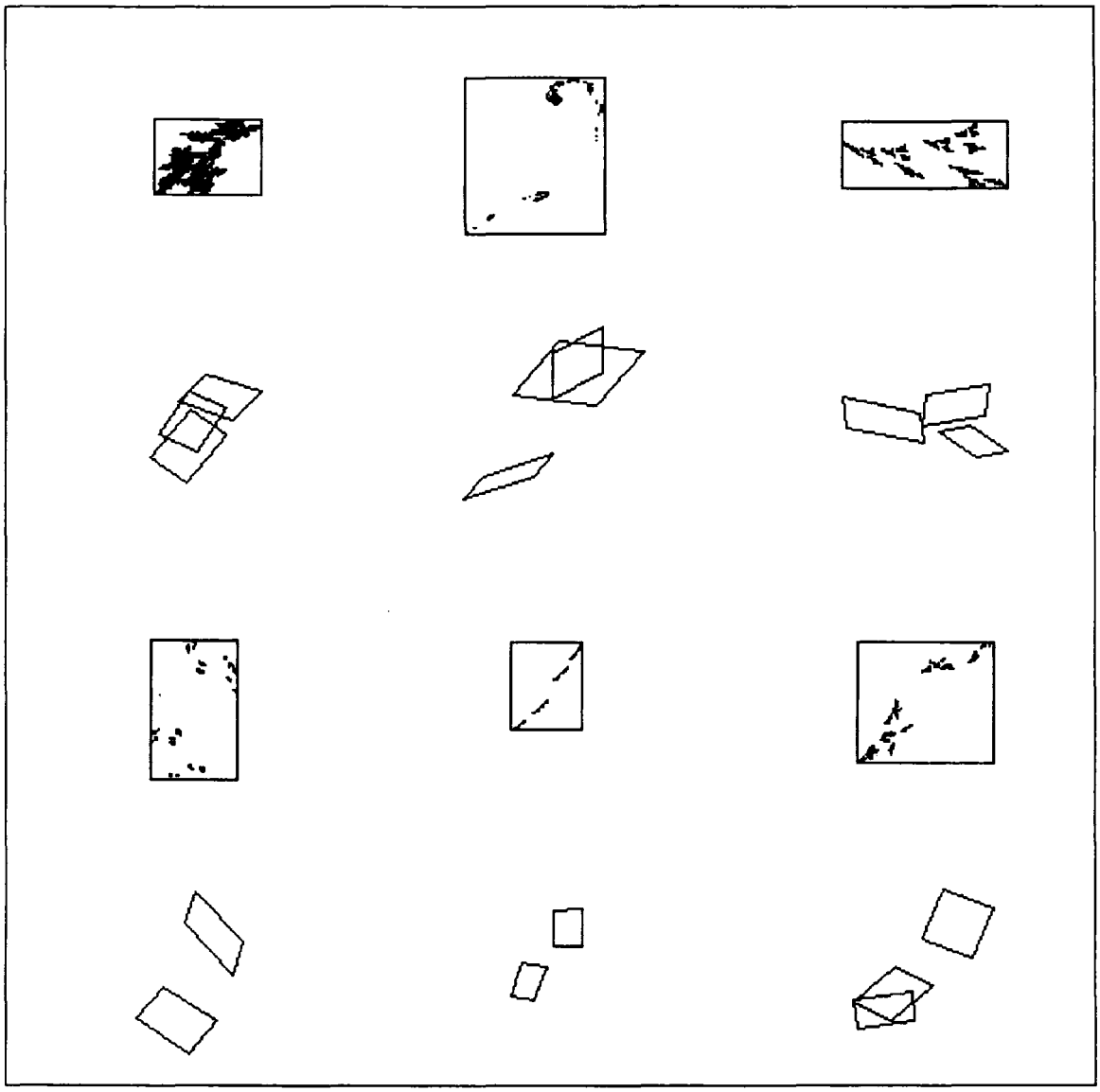


Figure 6.2 *The best-so-far solutions for encodings of the six randomly generated shapes of the test set using fitness function $\mathcal{B}$. Again, the layout is the same as that for figure B.2 in appendix B so that a direct comparison with the original shapes can be made.*

The quality of these encodings is in general very poor and especially so for the random fractals. One reason for this as demonstrated well by the attempted encodings for the fern and square shapes. The fitness function is simply too greedy, and insensitive to shape structure. For shapes such as the square where a slight modification in one of the mappings will make little difference to the number of points it covers, once solutions start to appear for which the number of pixels is near the maximum, little extra improvement is then made. This is seen clearly in graph C.28 which shows a good rate of average fitness increase due to the initial ease with which mappings can be made to cover the shape, but convergence is almost complete by the 75th generation. A similar trend is shown in the graph for the fern (C.27) but here the problem is aggravated by the tapering of the shape which results in the mappings clustering in regions of high point density. It is the more distributed nature of the points of the random fractals that result in their poor representations, with mappings migrating towards high density areas and neglecting the overall form of the shape.

The tendency for mappings to cluster at high point densities is not the only factor in the poor performance of function $\mathcal{B}$. By examining the dimensions of the collage mappings in figures 6.1 and 6.2 it becomes apparent that there is a tendency for the mappings to be small and square, that is, the contractivity of the collages are kept low and in general $r_1 \approx r_2$, $\theta_1 \approx \theta_2$, and the mappings approximate similitudes, thus preserving the interior angles of the bounding rectangle. The reasons for these two effects are closely linked and easily explained by considering what happens when a relatively good solution occurs in an early generation. Bearing in mind that function $\mathcal{B}$ rewards low contractivity and high shape coverage, it is likely that an good early solution will consist of some small mappings which all map onto the shape. The solution thus has a high fitness

and is allocated an above average number of offspring which find it easy to improve upon their parents fitness value by simply expanding the area of the shape covered by each mapping. This is possible since the area of a mapping is given by:

$$Area(w(\mathbf{S})) = r_1 r_2(\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2) Area(\mathbf{S}).$$

and the contractivity factor for the whole collage is simply the maximum $|r|$ value of any of the mappings. Thus each mapping can increase its values of r_1 and r_2 up to the contractivity factor without imposing any negative effect on the fitness of the whole collage. In fact, such action is almost guaranteed to increase the fitness of the collage since a larger area is extremely likely to cover more shape points. Hence it is understandable that $r_1 = r_2 = s$ frequently occurs. The reason that similitudes are favoured is that the $\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2$ term of the area equation is a maximum for $\theta_1 = \theta_2$ and so again the area of each mapping can be increased without any negative effects. Naturally, this process of improvement takes place over many generations, but there is a good chance that these continually improving solutions will always be above average and will eventually dominate the whole population with the result that by the time the limits of improvement have been reached there is insufficient genetic variation for further general improvement and hence the poor results.

Of all the encoding attempts with function $\mathcal{B}$ perhaps the best is that obtained for the Sierpinski triangle since, although the attractor is visually poor, the collage is close to one of the optimal. The reason for this above average performance is clear in the light of the above explanation. The Sierpinski triangle is a 'compact' shape in the sense that it has no thinly tapering parts or jutting peninsulas, and has a uniform point density over its whole surface. This alone would probably be enough to ensure that its results were above average, but we

must also consider that its collage consists of three 'square' mappings which are identical apart form their translational components, and each of which has three-fold rotational symmetry. The effect of this is that there are at least 27 distinct, perfect, three mapping collages. Hence in this case the preference for centralised square mappings is beneficial and the number of optimum collages improves the chances of getting a near optimal result.

As a demonstration of the search processes that occur during the running of the program, the next figure shows a sequence of the best-so-far attractors found using function $\mathcal{B}$ on the Sierpinski triangle. They represent a sample of the thirty-nine best-so-far solutions that were generated and, to demonstrate the convergence better, more examples of early results are shown since the last few attractors are all very similar. Also, as a check on whether the poor performances of $\mathcal{B}$ might be a flaw in the program, we used a population size of five hundred with the expectation of much improved results.

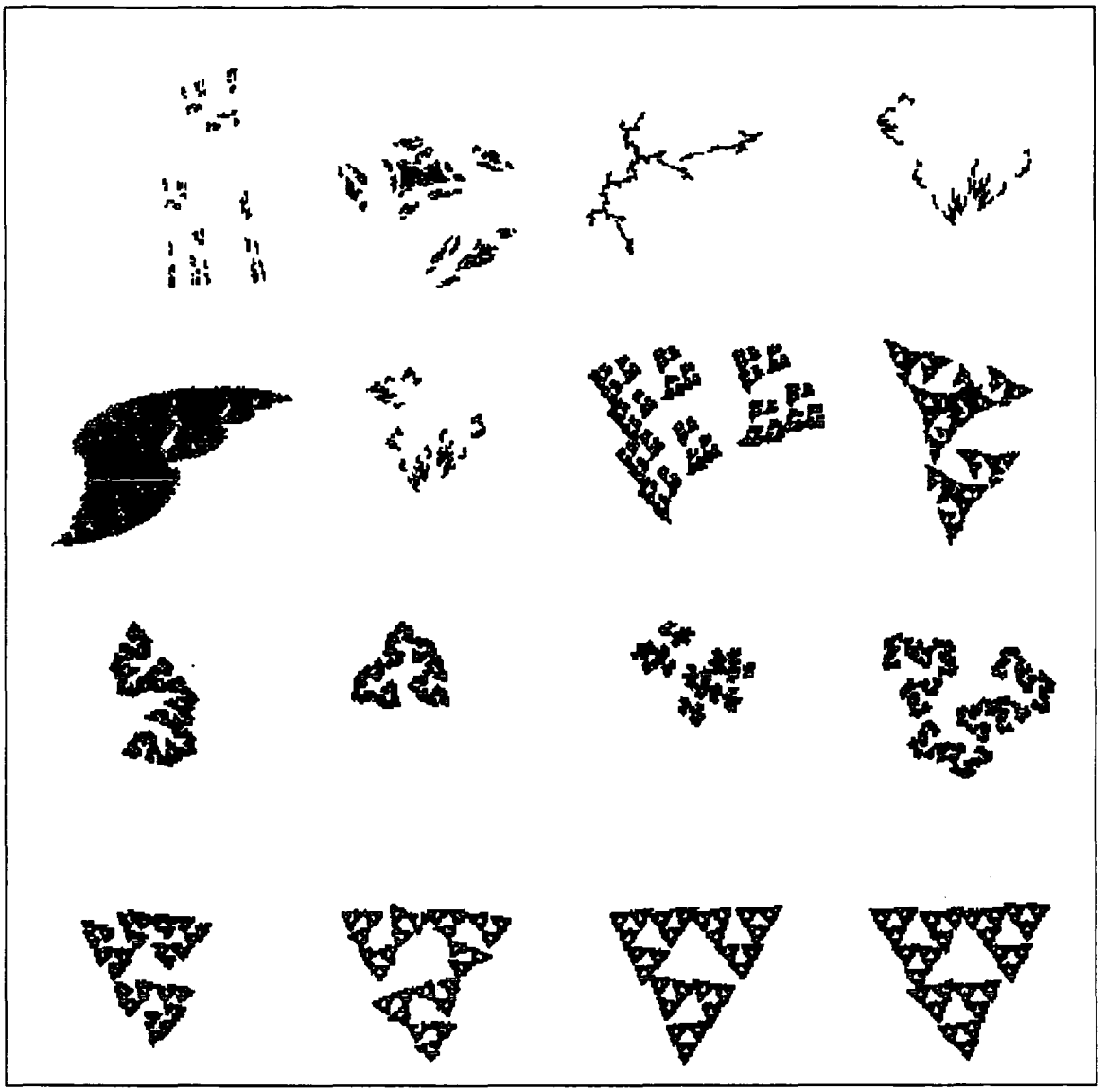


Figure 6.3 *A sequence from the set of best-so-far solutions generated during a run of the GA using fitness function $\mathcal{B}$, the Sierpinski triangle test shape, and a population size of five hundred.*

A final unexpected result obtained with function $\mathcal{B}$ is the production of collages for some of the random fractals with fitnesses better than could be achieved by the optimal solution. In retrospect however, this can be seen as another effect of the tendency of mappings to prefer areas of high point density where they can cover the majority of points yet remain small. This is demonstrated by the result for the fifth random fractal in figure 6.2. If it is compared with the ideal collage in figure B.2 it can be seen that the two small mappings correspond with the centres of the streamer-like halves of the shape and thus cover a high percentage of points whilst the collage has a contractivity factor of only 0.469. The ideal collage naturally covers all points but consists of two large mappings with a contractivity factor of 0.882 and so has a maximum fitness of 0.118. The best-so-far solution actually covered just 30% of the whole shape but this is enough to give it a fitness of 0.159.

We now turn attention to fitness function $\mathcal{A}$ which incorporates the evaluation of the Hausdorff metric. The results of using this function on five shapes of the test set can be seen in the following figure and graphs C.32-C.36.

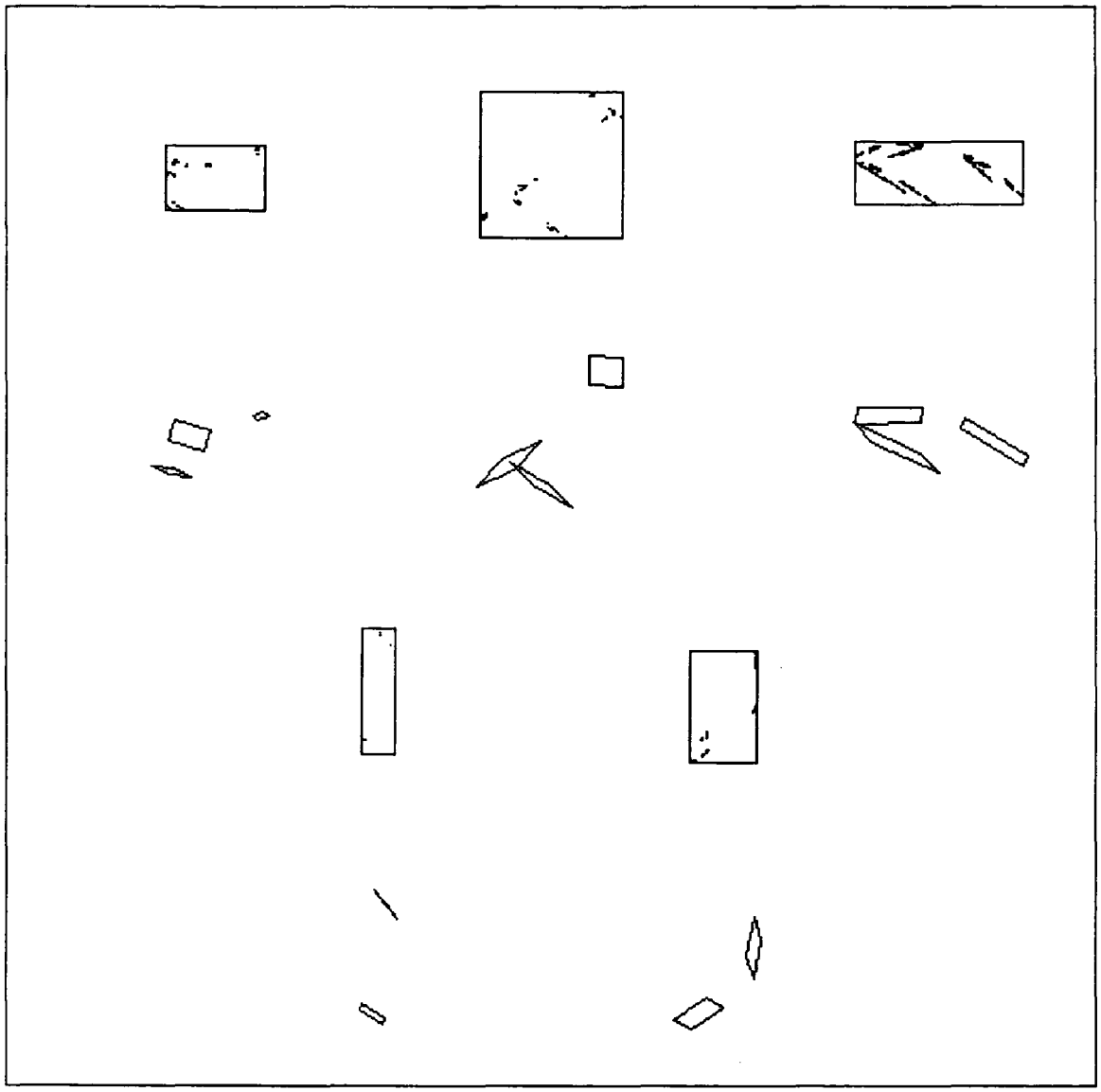


Figure 6.4 *The results of encoding (from top left to bottom right) the Sierpinski triangle, rand2, rand3, rand4, and rand5 using fitness function A. The choice of shapes was based on the number of points contained in each in order to minimise runtimes and also on which were good comparisons for the results of function B.*

Paradoxically, the more sophisticated fitness function gives worse results than those obtained previously. In general the collages consist of very small mappings distributed evenly over the shapes and resulting in sparse, poor quality attractors. The average fitness plots show a tendency for rapid convergence with little possibility of improved results coming from extended runs. The reason for these results would appear to be the form of the Hausdorff metric itself. As its definition reveals its value is ultimately just the distance between a pair of points, with one in the input shape and the other in the collage, or more specifically, in one mapping of the collage. Thus we get bad results similar to those of function $\mathcal{B}$ but for the opposite reason. It is now possible that the offspring of good solutions can improve their fitness by reducing the sizes of their constituent mappings since there is a good chance that this will not affect the one collage point that is determining the value of the Hausdorff distance, but is guaranteed to decrease the contractivity factor and hence improve fitness. Again, by the time the limit of this process has been reached, genetic variation has been lost and suboptimal solutions result. Since the area of each mapping is now under pressure to be minimised there is less requirement for similitudes and this is confirmed by the presence of more skewed and elongated mappings in figure 6.4. Although the mapping size is clearly the dominant factor the effect of the Hausdorff term is apparent by the positioning of the mappings on the shapes. This is shown well in the results for the second, fourth and fifth random fractals of figure 6.4 where the centres of the mappings in the best-so-far solutions correspond well with those of the optimal solutions. The effect of using $\mathcal{A}$ is not the generation of shape filling collages as might be expected considering the derivation of the function, but something akin to a minimum spanning of the input shape by MAP number of points.

The opposite pressures applied by functions $\mathcal{B}$ and $\mathcal{A}$ are demonstrated clearly by the results for the Sierpinski triangle. As just mentioned this shape gives the best result for function $\mathcal{B}$ but as figure 6.4 shows probably the worst result for $\mathcal{A}$. Graph C.38 indicates an extremely early and rapid convergence confirmed by the best-so-far result after one hundred generations having been found during the sixteenth.

The definition of function $\mathcal{C}$ is obviously an attempt at combining the two previous functions to balance their opposed forces. Results obtained are shown in the figure on the following page and in appendix C, graphs C.37-C.41.

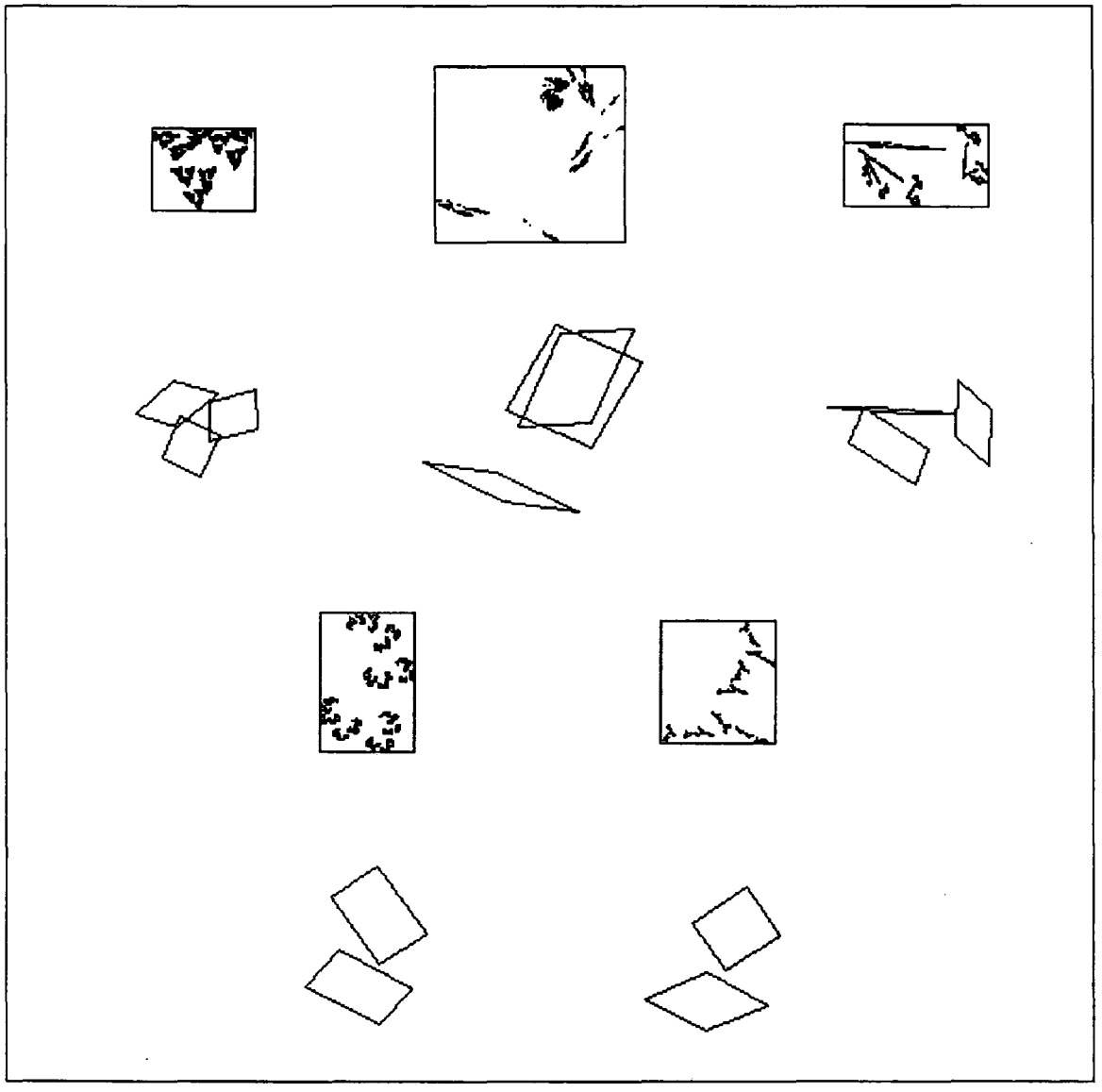


Figure 6.5 *Results obtained using fitness function C for the same test shapes as in figure 6.4.*

These results are an improvement on those of both previous sets although in absolute terms the quality of representation is still not good. The reappearance of large square mappings indicates that of the two competing terms the contractivity is stronger although the effects of the Hausdorff distance are apparent by their better distribution. The graphs show higher rates of average fitness growth, and in some cases sustain them up to the last generation indicating the were more computer resource available, better results would follow. As a test of this the following figure shows the collage and attractor obtained for the second random fractal using a population size of one thousand and a runtime of three hundred generations. Although the result is a great improvement on any yet achieved, (see the following figure and graph C.42), the runtime for the program was approximately sixty hours.

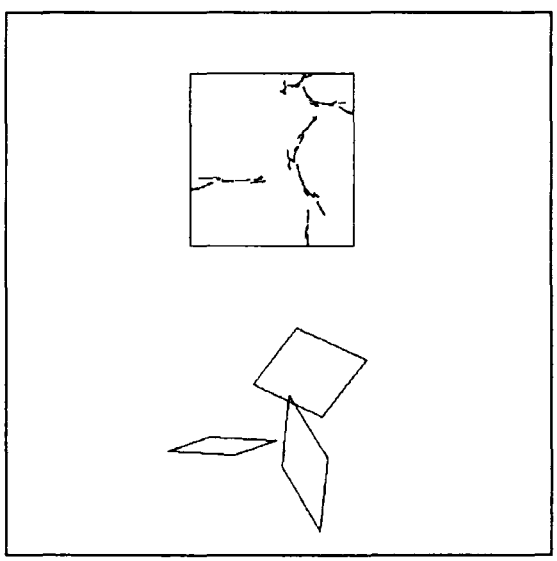


Figure 6.6 *The representation obtained for the second random fractal using fitness function C, a population size of one thousand and a runtime of three hundred generations.*

Since the poor performance of fitness functions $\mathcal{A}$ and $\mathcal{B}$ is in part attributed to their contractivity terms it could be suggested that this be dropped from their definitions. However, this would allow collages consisting of large mappings with contractivity factors approaching unity. It is easy to see that any large mappings that are not penalised for their size will be able to achieve high fitness values under any of the fitness functions considered and will hence quickly dominate the population. This is what in fact happens when the contractivity factors are removed from the fitness measures, with the best-so-far solution invariably including one of more near 'full size' mappings. Although the collages accurately cover the shapes, the attractors are still poor because of the very high contractivity factors.

6.9 Conclusion

The results of this chapter have demonstrated the use of a GA in the calculation of fixed size IFS encodings of whole two-dimensional shapes. No restrictions were placed upon the type of shapes used, with both complex natural forms and figures of Euclidean geometry being present in the set of test shapes upon which the program performance was evaluated. With reference to this specific application we can draw the following conclusions:

1. Once suitable program parameters are set to give best results within the constraints imposed by computer resource limitations, the form of the fitness function becomes of critical importance determining through its discriminatory power the speed of population convergence and ultimately the value of the best-so-far solution.

2. The term derived in the collage theorem which relates the distances of a collage and its associated attractor from the initial shape is inadequate in itself as a fitness function.

3. The presence of a contractivity term in the fitness function is necessary to prevent the use of large mappings and the subsequent weak bound that is placed upon the quality of the attractor.

4. The presence of a metric distance term in the fitness function is necessary to provide the required degree of discrimination for cases where the input shape consists of widely distributed and sparse points.

5. The use of the Hausdorff distance as the metric requires that the fitness function include a greedy term which rewards collages with high shape coverage.

As for the reproductive plan D_1, its has demonstrated high quality performance by producing smoothly increasing average population fitnesses for correctly set program parameters. The use of non-arbitrary offspring allocation, and the guaranteeing of at least one population share to each solution, has been shown to have the desired effect of maintaining population diversity, whilst the use of an $r = 2$ sample size has not adversely effected program performance, and has enabled a highly efficient implementation.

In terms of the general practicality of using a GA for collage construction, we have shown that the program is basically $O(n^2)$ in the number of pixels contained in the shape but that the expense of evaluating the Hausdorff distance and the size of typical shapes results in extended runtimes. For example, using fitness function C which incorporates the properties described above, average runtimes for one hundred generations using a population size of one hundred solutions were of the order of eight hours, whilst sixty hours of processor time were required for

the attainment of results of a qualitatively high standard. The length of such runtimes and the stochastic nature of a GA which obstructs exact program analysis renders the extensive investigation of encoding performance impractical with the current implementation.

Despite the current speed and accuracy limitations we believe that the basic principle of using a GA for the construction of full shape collages is sound, and that no theoretical barriers exist to the future development of the present implementation and its ultimate incorporation into a machine vision system as part of a shape representation scheme.

7 CONCLUSION

We began with a brief discussion of the requirements of a general machine vision system, concentrating in particular on the representation schemes that such a system could be expected to employ, based upon a study of the available psychophysical data. Following this and a review of the representation schemes employed by past systems, we suggested the use of iterated function systems as a fusion of conventional geometric and pictorial schemes.

After a presentation of the theory of IFSs we introduced a formal framework within which to construct a two-dimensional shape representation scheme and derived the theoretical properties such a scheme would possess, giving examples. We then tackled the fundamental problem of obtaining automatic shape encodings.

The work of chapter four on an encoding scheme based upon the simplification of concentrating on a shapes boundary led to the conclusion that, whilst the program developed may have applications in computer graphics, the limitations on the range of shapes that could be successfully encoded, and the inescapable requirement of constructing full shape collages, meant that the implementation was inadequate as a basis for a general shape representation scheme.

An adaptive algorithm was identified as necessary to cope with the complex search domain presented by the problem of full collage generation, and a specific

type, the genetic algorithm, was selected. A discussion of the theoretical properties and practical limitations of GAs was presented in chapter five, culminating in the introduction of a new reproductive plan, D_1, designed to improve program performance and increase implementation efficiency.

An implementation of a GA incorporating a highly efficient version of D_1 was described in chapter six, and its performance in automatically calculating full collages for general shape input was assessed. The conclusions of this chapter gave guidelines to the requirements of a GA intended for IFS encoding, including the importance of the form of the fitness function and the characteristics that it is required to display. The cost of evaluating the Hausdorff metric for typical sets was found to be prohibitive using the current hardware, resulting in extended program runtimes. However, the general conclusion was that the application of a GA to the collage construction problem is fundamentally sound.

In general we conclude that iterated function systems have the potential to form the foundation of a powerful shape representation scheme, combining the positive attributes of conventional geometric and pictorial models used in contemporary machine vision systems. The GA encoding technique has been shown to be a possible basis for a practical implementation of such a scheme, with the capability of automatically generating shape encodings.

7.1 Research Directions

The work presented here suggests the following areas for future research:

1. The extended runtimes produced by the current implementation are an effect of the large amounts of data that must typically be processed and not the

inherent complexity of the algorithm itself, which is basically $O(n^2)$ with respect to the number of data points. Therefore, the conclusions of chapter six can be confirmed by the use of faster hardware so enabling more exhaustive investigation of program performance. The fundamental nature of a GA and the minimal evaluation and communication required by the reproductive plan D_1 suggest that the current program would transfer well onto a parallel architecture. An order of magnitude increase in encoding speed would make the assessment of the program based on code fidelity a practical proposition. A further order of magnitude increase would result in encoding times down to a matter of minutes at which point machine vision applications would start to become practical.

2. The reproductive plan D_1 avoids the need for arbitrary scaling of offspring numbers and, by requiring only the relative evaluation of a subset of the whole population, enables highly efficient implementations. It therefore has possible application to genetic algorithms in general and as such warrants further evaluation, especially with reference to the effects of varying the sample size. Even if D_1 is proved to have performance not significantly different from more conventional reproductive plans, its improved efficiency and integrity will prove highly valuable.

3. From the definitions of chapter two it is apparent that the theory of IFSs is applicable to n-dimensional spaces, and thus the logical progression of the current work is the extension of the encoding technique to three-dimensional forms, thereby permitting the direct modelling of real world objects. This can be achieved by reformulating the formalism of chapter three for the space $\mathcal{H}(\mathbf{R}^3)$. This will not however affect the properties of the representation such as stability and robustness since these are inherent in IFSs of any dimension. The RIA algorithm can be simply extended to the three-dimensional case thus preserving the ease of model manipulation. Finally, the GA described in chapter six can, in

theory, be simply modified to handle three-dimensional input, the only problems foreseen being those of increased search domain complexity and the difficulty of working with three dimensional data.

REFERENCES

Aho A.V., Hopcroft J.E., Ullman J.D., *Data Structures and Algorithms*, Addison-Wesley, 1983.

Anderson J.A.D.W., Sullivan G.D., Baker K.D., "Constrained Constructive Solid Geometry: A Unique Representation of Scenes", *Proceedings: Fourth Alvey Vision Conference, Manchester*, **91–96**, (August 1988).

Arbib M.A., Hanson A.R., "Vision, Brain, and Cooperative Computation: An Overview", in *Vision, Brain and Cooperative Computation*, ed. M.A.Arbib and A.R.Hanson, (1988).

Asada H., Brady M., "The Curvature Primal Sketch", *IEEE Transactions on Pattern Analysis and Machine Intelligence vol. PAMI-8, no.1*, **2–14**, (January 1986).

Bailey T., Cowles J., "A Convex Hull Inclusion Test", *IEEE Transactions on Pattern Analysis and Machine Intelligence vol. PAMI-9, no.2*, **312–317**, (March1987).

Bajcsy R., Solina F., "Three Dimensional Object Representation Revisited", *IEEE First International Conference on Computer Vision*, **230–240**, (1987).

Barnsley M.F., Demko S., "Iterated Function Systems and the Global Construction of Fractals", *Proceedings of the Royal Society of London A399*, **243–275**, (1985).

Barnsley M.F., Ervin V., Hardin D., Lancaster J., "Solution of an Inverse Problem for Fractals and Other Sets", *Proceedings of the National Academy of Science USA vol.83*, **1975–1977**, (April 1986).

Barnsley M.F., Sloan A.D., "A Better Way to Compress Images", *BYTE*, **215–223**, (January 1988).

Barnsley M.F., *Fractals Everywhere*, Academic Press, 1988.

Barnsley M.F., "Fractals and Chaos", *Proceedings of the BCS seminar on Fractals and Chaos*, **1–21**, (December 1989).

Booker L., "Improving Search in Genetic Algorithms", in *Genetic Algorithms and Simulated Annealing*, ed. L.Davis, (1987).

Brooks R.A., "Symbolic Reasoning among 3D Models and 2D Images", *Artificial Intelligence 17*, **285–348**, (1981).

Bullock B.L., "The Necessity for a Theory of Specialized Vision", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

DeJong K.A., *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, PhD. Thesis, University of Mitchigan, 1975.

DeJong K.A., "Learning with Genetic Algorithms: An Overview", *Machine Learning, 3*, **121–138**, (1988).

Fischler M.A., "On the Representation of Natural Scenes", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Fisher R., "SMS: A Suggestive Modelling System for Object Recognition", *Image and Vision Computing vol.5 no.2*, **98–104**, (May 1987).

Fitzpatrick M.J., Grefenstette J.J., "Genetic Algorithms in Noisy Environments", *Machine Learning, 3*, **101–102**, (1988).

Funt B.V., "Problem-Solving with Diagrammatic Representations", *Artificial Intelligence 13(3)*, **201–230**, (1980).

Giles P.A., Purvis A., Waugh D., Garigliano R., "Iterated Function Systems and 2-D Shape Representation", *Proceedings: Fifth Alvey Vision Conference, Reading*, **49–53**, (September 1989).

Hanson A.R., Riseman E.M., "VISIONS: A Computer System for Interpreting Scenes", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Hayes P.J., "The Second Naïve Physics Manifesto", in *Formal Theories of the Commonsence World, Ablex Publishing Corp.*, ed. J.R.Hobbs, R.C.Moore, (1985).

Holland J.H., *Adaptation in Natural and Artificial Systems*, University of Mitchigan Press, 1975.

Horn A.N., "IFSs and the Interactive Design of Tiling Structures", *Proceedings of the BCS seminar on Fractals and Chaos*, **22–39**, (December 1989).

Koons D.B., McCormick B.H., "A Model of Visual Knowledge Representation", *IEEE First International Conference on Computer Vision*, **365–372**, (1987).

Kosslyn S.M., Schwartz S.P., "Visual Images as Spatial Representations in Active Memory", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Levy-Vehel J., Gagalowicz A., "Shape Approximation by a Fractal Image", *Eurographics 1987, North Holland*, **159–180**, (1987).

Libeskind-Hadas R., Maragos P., "Application of Iterated Function Systems and Skeletonization to Synthesis of Fractal Images", *Visual Communication and Image Processing II, SPIE vol.845*, **277–285**, (1987).

Mandelbrot B.B., *The Fractal Geometry of Nature*, W.H. Freeman and Co., 1982.

Marr D., "Representing Visual Information", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Marr D., Nishihara K., "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes", *Proceedings of the Royal Society of London B200*, **269–294**, (1978).

Mokhtarian F., Mackworth A., "Scale-Based Description and Recognition of Planar Curves and 2-D Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence vol. PAMI-8, no.1*, **34–43**, (January 1986).

Mumford D., "The Problem of Robust Shape Descriptors", *IEEE Ist International Conference on Computer Vision*, **602–606**, (1987).

Nevatia R., "Characterization and Requirements of Computer Vision Systems", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Pentland A.P., "Fractal-Based Description of Natural Scenes", *IEEE Transactions on Pattern Analysis and Machine Intelligence vol. PAMI-6, no.6*, **661–674**, (November 1984).

Pentland A.P., "Perceptual Organization and the Representation of Natural Form", *Artificial Intelligence 28*, **293–331**, (1986).

Pentland A.P., "Recognition by Parts", *IEEE First International Conference on Computer Vision*, **612–620**, (1987).

Tsotsos J.K., "Knowledge and the Visual Process: Content, Form, and Use", *Pattern Recognition, vol. 17, no. 1*, **13–27**, (1984).

Tsotsos J.K., "A Complexity Level Analysis of Vision", *IEEE First International Conference on Computer Vision*, **346–355**, (1987).

Waugh D.A., "Assessment and Comparison of Existing Approaches to Computer Vision", *Internal Report, Dept. Computer Science, Durham University*, **1–24**, (August 1989).

Weisstein N., Maguire W., "Computing the Next Step : Psychophysical Measures of Representation and Interpretation", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

BIBLIOGRAPHY

Aloimonos J.Y., Shulman D., "Learning Shape Computations", *Proceedings: DARPA Image Understanding Workshop*, **862–873**, (February 1987).

Arbib M.A., Hanson A.R., (eds.), *Vision, Brain and Cooperative Computation*, MIT Press, 1988.

Aviad Z., Lozinskii E., "On a Conceptual Description of Images", *Pattern Recognition Letters, North Holland*, **51–57**, (1985).

Aviad Z., "A Discrete Scale-Space Representation", *IEEE First International Conference on Computer Vision*, **417–421**, (1987).

Baker J.E., "Reducing Bias and Inefficiency in the Selection Algorithm", *Proceedings: Genetic Algorithms and Their Applications - The Second International Conference on Genetic Algorithms*, **14–21**, (1987).

Ballard D.H., Brown C.M., *Computer Vision*, Prentice Hall, 1982.

Barrow H.G., Tenenbaum J.M., "Recovering Intrinsic Scene Characteristics from Images", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Beard.N., "Ratio Revolution", *Image Processing*, **16–17**, (May/June 1990).

Bennett B.M., Hoffman D.D., Prakash C., "Perception and Computation", *IEEE First International Conference on Computer Vision*, **356–364**, (1987).

Besl P.J., Jain R.C., "Three-Dimensional Object Recognition", *ACM Computing Surveys, vol.17, no.1*, **74–145**, (March 1985).

Bixler J.P., Watson L.T., Sanford J.P., "Spline-Based Recognition of Straight Lines and Curves in Engineering Line Drawings", *Image and Vision Computing, vol.6, no.4*, **262–269**, (1988).

Brisdon K., Sullivan G.D., Baker K.D., "Feature Aggregation in Iconic Model Evaluation", *Proceedings: Fourth Alvey Vision Conference, Manchester*, **19–22**, (August 1988).

Chein C.H., Aggarwal J.K., "Shape Recognition from Single Silhouettes", *IEEE First International Conference on Computer Vision*, **481–490**, (1987).

Cyganski D., Orr J.A., Cott T.A., Dodson R.J., "Development, Implementation, Testing and Application of an Affine Invariant Curvature Function", *IEEE First International Conference on Computer Vision*, **496–500**, (1987).

Dawkins R., *The Selfish Gene*, Oxford University Press, 1976.

Dawkins R., *The Extended Phenotype*, Oxford University Press, 1982.

Dewdney A.K., "Mathematical Recreations - How to Transform Flights of Fancy into Fractal Flora or Fauna", *Scientific American*, **90–93**, (May 1990).

Ettinger G.J., "Large Hierarchical Object Recognition using Libraries of Parameterized Model Sub-Parts", *Proceedings of the Computer Society*

Conference on Computer Vision and Pattern Matching, Ann Arbor, Michigan, **32–41**, (June 1988).

Gleick J., *Chaos*, Cardinal Press, 1987.

Gregory R.L., *The Intelligent Eye*, McGraw-Hill, New York, 1970.

Lamdan Y., Schwatz J.T., Wolfson J., "Object Recognition by Affine Invariant Matching", *Proceedings of the Computer Society Conference on Computer Vision and Pattern Matching, Ann Arbor, Michigan*, **335–344**, (June 1988).

Leavers V., "Use of the Radon Transformation as a Method of Extracting Symbolic Representations of Shape in Two Dimensions", *Proceedings: Fourth Alvey Vision Conference, Manchester*, **273–279**, (August 1988).

Levine M.D., "A Knowledge-Based Computer Vision System", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Osbourn G.C., "A New Approach to Machine-Based Perception of Monocular Images", *Proceedings: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, Mitchigan*, **864–870**, (June 1988).

Peitgen H.O., Richter P.H., *The Beauty of Fractals*, Springer-Verlag, 1988.

Reynolds G., Beveridge J.R., "Searching for Geometric Structure in Images of Natural Scenes", *Proceedings: DARPA Image Understanding Workshop*, **257–271**, (February 1987).

Ridley M., *The Problems of Evolution*, Oxford University Press, 1983.

Shamos M.I., Hoey D., "Closest-Point Problems", *Proceedings: 16th. Annual Symposium Of the Foundation for Computer Science*, **151–162**, (1975).

Tanimoto S.L., "Regular Hierarchical Image and Processing Structures in Machine Vision", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

Teh C., Chin R.T., "A Scale-Independant Dominant Point Detection Algorithm", *Proceedings of the Computer Society Conference on Computer Vision and Pattern Matching, Ann Arbor, Michigan*, **229–234**, (June 1988).

Todd H.S., "A Descriptive Pattern Recognition System Applied to Pictorial Patterns where the Discriminating Information is carried in the Object Shape", *Proceedings: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, Mitchigan*, **430–436**, (June 1988).

Van Hove P., "Model-Based Silhouette Recognition", *Proceedings: IEEE Computer Society Workshop on Computer Vision*, **88–93**, (1987).

Wang Y.F., Magee M.J., Aggarwal J.K., "Matching 3-D Objects using Silhouettes", *IEEE Transactions on Pattern Analysis and Machine Intelligence vol. PAMI-6, no.4*, **513–518**, (July 1984).

Weiss I., "Curve Fitting using a Varying Mesh", *Proceedings: IEEE Computer Society Workshop on Computer Vision*, **311–314**, (1987).

Wolfson H., "On Curve Matching", *Proceedings: IEEE Computer Society Workshop on Computer Vision*, **307–310**, (1987).

Zucker S.W., "Vertical and Horizontal Processes in Low-Level Vision", in *Computer Vision Systems*, ed. A.Hanson and E.Riseman, (1978).

GLOSSARY OF TERMS

ACRONYM - a vision system incorporating model based reasoning and a generalised cylinder object representation scheme.

Adaptive Algorithm - a class of algorithm in which a set of structures is iteratively updated, and of which a genetic algorithm is an example.

Arc - a section of a shapes bounding contour produced by segmentation at curvature zero crossings.

Attractor - the limit point of an iterated function system.

Collage - the name given to any shape specific set of contraction mappings for which the union of the mappings is equal to the shape itself.

Crossover - an operator on the population of a genetic algorithm which exchanges information between solutions.

CSG - an acronym for Constructive Solid Geometry, a geometric representation scheme in which an object is described by the Boolian combination of volumetric shape primitives.

$\mathbf{D_1}$ - a reproductive plan that avoids arbitrary allocation of offspring and allows efficient implementation of genetic algorithms.

Fitness Function - a measure of the value of each solution in the population of a genetic algorithm.

GA – an abbreviation for Genetic Algorithm, a search algorithm based upon a simple model of population genetics.

GEN – a program parameter specifying the number of generations for which a genetic algorithm is to be run.

Generalised Cylinder – The volume described by a cross-sectional area of fixed shape but varying size as it is swept along a space curve. Used as a volumetric shape primitive in some geometric representation schemes.

Generation – a time period equal to one iteration of a genetic algorithm. Also used to refer to the population during such a time period.

Glimpse – a vision system incorporating pictorial representations or 'glimpses' of objects.

IFS – an abbreviation for Iterated Function System, a set of contraction mappings on a metric space which, when applied iteratively to an any subset of the space, always produce the same subset in the limit.

MAP – a program parameter for the implementation of a genetic algorithm that specifies the number of mappings to be used in the collages.

Matching Set – a set of arcs used to find mappings onto the boundary of a shape.

MUT – a program parameter specifying the mutation probability in a genetic algorithm.

Mutation – an operator on the solutions in the population of a genetic algorithm that introduces random changes.

Offspring – the children allocated to a solution on each iteration of a genetic algorithm.

POP – a program parameter specifying the population size for a genetic algorithm.

Population – the set of solutions (structures) modified by a genetic algorithm.

Population Share – the number of times a solution contributes genetic material to the next generation. Since each child has two parents, there are $2N$ share allocations in each population of size N.

Reproductive Plan – the rules controlling the allocation of offspring and the use of genetic operators in a genetic algorithm.

RIA – an abbreviation for Random Iteration Algorithm, used for obtaining the attractor of an iterated function system.

Schemata – bit patterns treated as formal random variables for the purpose of analysis the behaviour of genetic algorithms.

Shape – defined to be any compact subset of the Euclidean plane.

SMS – an acronym for Suggestive Modelling System, an object representation scheme motivated by the need for 'visual salient' primitives.

Solution - one of the trial solutions that constitute the population of GA. More formally a structure that represents a solution in some search domain which undergoes iterative adaptation.

String - a string of binary digits that represent a trial solution in a GA.

Structure - a data element that undergoes successive modification in an adaptive system.

Superquadrics - a family of three-dimensional shapes defined by the surface swept out by the tip of the parameterised vector $\mathcal{X}(\nu,\omega)$, where ν, and ω are latitudinal and longitudinal angles respectively. Used as the primitives in some CSG implementations.

VISIONS - a vision system based on rule based knowledge representation.

WHISPER - a vision system incorporating diagrammatic reasoning.

XLN - a program parameter specifying the maximum crossover length in a genetic algorithm.

APPENDIX A

We present here the C object code for the genetic algorithm described in chapter six. The text in normal type is explanatory material and is not part of the code itself. C key-words appear in bold type.

```
#include < stdio.h >
#include < math.h >
```

MACRO DECLARATIONS

MAP . . . . . number of mappings.

SUB . . . . . subsampling factor.

POP . . . . . population size.

GEN . . . . . number of generations.

MUT . . . . . mutation probability.

XLN . . . . . maximum crossover length.

```
#define MAP 3
#define SUB 1
#define POP 100
#define GEN 100
#define MUT 0.01
#define XLN (6 * MAP) - 2
```

FUNCTION DECLARATIONS

The explanation of the operation of each function can be found at the head of its definition.

```
int rand();
void medical();
void propagate();
void read_image();
void initialise();
void write_best();
```

GLOBAL VARIABLES

dx[ ], dy[ ] . . . . . extent of shape in x and y directions.

cen[ ] . . . . . centroid of shape.

pixels . . . . . the number shape pixels.

best . . . . . fitness of best-so-far solution.

colour . . . . . highest pixel value in array[ ][ ].

oldfit, newfit . . . . . pointers to fitness values.

data[ ] . . . . . powers of 2.

parent_fit[ ] . . . . . fitness of parents.

child_fit[ ] . . . . . fitness of children.

shape[ ][ ] . . . . . co-ordinates of shape pixels.

mapped[ ][ ] . . . . . co-ordinates of mapped pixels.

array[ ][ ] . . . . . array representation of image.

parents[ ][ ] . . . . . bit-string representation of generation (n).

children[ ][ ] . . . . . bit-string representations of generation (n+1).

oldgen, newgen . . . . . pointers to populations.

float $dx[2]$, $dy[2]$;

int $cen[2]$, $pixels$;

int $best = 0$, $colour = 1$;

int $*oldfit, *newfit, *dummyfit$;

char $data[8] = \{1, 2, 4, 8, 16, 32, 64, 128\}$;

int $parent\_fit[POP]$, $child\_fit[POP]$;

int $shape[5000][2]$, $mapped[5000][2]$, $array[512][512]$;

char $parents[POP][6 * MAP]$, $children[POP][6 * MAP]$;

char $(*newgen)[6 * MAP]$, $(*oldgen)[6 * MAP]$, $(*dummygen)[6 * MAP]$;

MAIN

The program takes the following combination of parameters:

$$image\_file\ [population\_file].$$

IMAGE_FILE is the output file of the image processor and must contain the following information: the number of pixels in the shape; the position of the centroid of the shape in the image plane; and the extent of the shape in the x and y directions. POPULATION_FILE is optional. If present it must contain a population of solutions compatible with the current program parameters. No checks for compatibility are made. If the population file is not present then a random starting population is generated by INITIALISE(). The image file is read by the function READ_IMAGE(). Calculation of successive generations is handled by the function PROPAGATE().

main(argc,argv)

int $argc$;

char $**argv$;

```
{register int i,j;

if ((argc < 2) || (argc > 3)) {printf("incorrect argument number \n"); exit(0);}

read_image(argv[1]);
initialise(argv[2]);
oldgen = parents; oldfit = parent_fit;
newgen = children; newfit = child_fit;

for (i = 0; i < GEN; i++)
   {propagate();
   dummygen = oldgen; dummyfit = oldfit;
   oldgen = newgen; oldfit = newfit;
   newgen = dummygen; newfit = dummyfit;}}
```

READ_IMAGE

Reads image data from output file of image processor. Initialises the global arrays shape[ ][ ] and array[ ][ ]. The position values in the imagefile are scaled by 1024 to allow use of integer maths without loss of accuracy. Other values are scaled to match.

filename . . . . . file from which image data is read.

```
void read_image(filename)
char *filename;
{int x,y;
FILE *in;
register int i;
```

```
if ((in = fopen(filename,"r")) == NULL) {printf("image file error \n"); exit(0);}
fread(&pixels, 4, 1, in);
fread(&cen[0], 4, 1, in); fread(&cen[1], 4, 1, in);
fread(&dx[0], 4, 1, in); fread(&dx[1], 4, 1, in);
fread(&dy[0], 4, 1, in); fread(&dy[1], 4, 1, in);

for (i = 0; i < pixels; i++)
   {fread(&shape[i][0], 4, 1, in); fread(&shape[i][1], 4, 1, in);
   x = (cen[0] + shape[i][0])/1024; y = (cen[1] + shape[i][1])/1024;
   array[x][y] = colour;}

cen[0]* = 256; cen[1]* = 256;}
```

INITIALISE

Initialises the first population. If a filename is given then the initial population is read from that, otherwise a random starting population is generated.

filename . . . . . name of file containing initial population.

```
void initialise(filename)
char *filename;
{FILE *in;
register int i,j;

if (filename == NULL)
   {for (i = 0; i < POP; i++)
      {for (j = 0; j < 6 * MAP; j++)
      {parents[i][j] = (char) rand(-128,127);}}}
else if((in = fopen(filename,"r")) != NULL)
```

```
{for (i = 0; i < POP; i++)
   {for (j = 0; j < 6 * MAP; j++) fread(&parents[i][j], 1, 1, in);
   fread(&parent_fit[i], 4, 1, in);}
fclose(in);}
else {printf("population file error \n"); exit(0);}}
```

PROPAGATE

Produces a child for each 'mother' solution within the population. A pair of possible mates is chosen at random, and the fitter of the two is selected as the 'father'. A crossover point and length are chosen at random. The bits between the endpoints of the crossover are copied from the father into the mother to create the child, which is then placed in the next generation. Masks are created for the end bytes of the crossover so that crossover length is not restricted to a whole number of bytes. One bit of the child is altered with probability MUT.

mut . . . . . bit of string to be mutated.

mask . . . . . template for crossover.

mate . . . . . chosen mate.

```
void propagate()
{int mut;
char mask;
int mate, k, l, m;
register int i, j;

for (i = 0; i < POP; i++)
   {for (j = 0; j < 6 * MAP; j++) newgen[i][j] = oldgen[i][j];
   newfit[i] = 0;
```

$l = rand(0, POP - 1)$; **if** $(oldfit[l] == 0)$ medical(l);

$k = rand(0, POP - 1)$; **if** $(oldfit[k] == 0)$ medical(k);

$mate = (oldfit[l] > oldfit[k])\ ?\ l : k$;

$l = rand(0, (6 * MAP - 1))$; $k = rand(0, XLN)$;

for $(j = l;\ j < (l + k);\ j++)$ $newgen[i][j\%(6 * MAP)] = oldgen[mate][j\%(6 * MAP)]$;

$m = (l - (1 + 6 * MAP))\%(6 * MAP)$;

$mask = data[rand(0, 7)] - 1$;

$newgen[i][m] = (oldgen[mate][m]\ \&\ mask) + (oldgen[i][m]\ \&\ (\tilde{}mask))$;

$m = (l + k)\%(6 * MAP)$;

$mask = -data[rand(0, 7)]$;

$newgen[i][m] = (oldgen[mate][m]\ \&\ mask) + (oldgen[i][m]\ \&\ (\tilde{}mask))$;

if $(rand(0, 10000) < 10000 * MUT)$

$\{mask = data[rand(0, 7)]$;

$mut = rand(0, (6 * MAP - 1))$;

$newgen[i][mut] = (newgen[i][mut]\ \&\ (\tilde{}mask)) + ((\tilde{}newgen[i][mut])\ \&\ mask);\}\}\}$

MEDICAL

Evaluates the fitness of a structure in the current population. The bit-string is decoded into MAP mappings each with five coefficients, $s_x, s_y, \theta, x_0, y_0$. These values are then used to find the six parameters needed to define each affine mapping. When the fitness of a structure exceeds that of any previously found, the IFS it represents is output by the function WRITE_BEST.

n . . . . . the current structure.

smax . . . . . contractivity factor.

tmp[ ] . . . . . contains $s_x, s_y, \theta, x_0, y_0$ values.

map[ ] . . . . . mapping coefficients.

points . . . . . number of mapped points.

min . . . . . minimum distance between points.

max1, max2 . . . . . Hausdorff measures.

```
void medical(n)
int n;
{int x, y;
float smax;
register int j, k;
int map[MAP][6];
float tmp[8];
int dummy[2], points;
int min, max1, max2, dist;

for (j = 0, smax = 0.0; j < 6 * MAP; j+ = 6)
  {tmp[0] = (float) (oldgen[n][j]) * 1.414;
  if (tmp[0] > smax) smax = tmp[0];
  else if (-tmp[0] > smax) smax = -tmp[0];
```

```
tmp[1] = (float) (oldgen[n][j + 1]) * 1.414;
if (tmp[1] > smax) smax = tmp[1];
else if (-tmp[1] > smax) smax = -tmp[1];
tmp[2] = sin((float) (oldgen[n][j + 2]) * M_PI/128.0);
tmp[3] = cos((float) (oldgen[n][j + 2]) * M_PI/128.0);
tmp[4] = sin((float) (oldgen[n][j + 3]) * M_PI/128.0);
tmp[5] = cos((float) (oldgen[n][j + 3]) * M_PI/128.0);
tmp[6] = ((float) (oldgen[n][j + 4]) + 128.0) * (dx[0] - dx[1]) + 255.0 * dx[1];
tmp[7] = ((float) (oldgen[n][j + 5]) + 128.0) * (dy[0] - dy[1]) + 255.0 * dy[1];

map[j/6][0] = (int)(tmp[0] * tmp[3]);
map[j/6][1] = (int)(-tmp[1] * tmp[4]);
map[j/6][2] = (int)(tmp[0] * tmp[2]);
map[j/6][3] = (int)(tmp[1] * tmp[5]);
map[j/6][4] = (int)(tmp[6] * 1024.0);
map[j/6][5] = (int)(tmp[7] * 1024.0);}

for (k = 0, max2 = points = oldfit[n] = 0; k < MAP; k++)
  {for (j = 0; j < pixels; j+ = SUB)
    {x = (map[k][0] * shape[j][0] + map[k][1] * shape[j][1] + map[k][4] + cen[0])/262144;
    y = (map[k][2] * shape[j][0] + map[k][3] * shape[j][1] + map[k][5] + cen[1])/262144;
     if ((array[x][y] > 0) && (array[x][y] <= colour))
       {oldfit[n]++; array[x][y] = colour + 1;
       mapped[points][0] = x * 1024 - cen[0]/256;
       mapped[points][1] = y * 1024 - cen[1]/256;
       points++;}
     else if ((array[x][y] <= 0) && (array[x][y] > -colour))
       {array[x][y] = -colour;
```

```
        mapped[points][0] = x * 1024 - cen[0]/256;
        mapped[points][1] = y * 1024 - cen[1]/256;
        points++;}}}

for (k = 0, max1 = 0; k < pixels; k+= SUB)
  {for (j = 0, min = 262144; j < points; j++)
    {dist = (shape[k][0] - mapped[j][0]) * (shape[k][0] - mapped[j][0])/262144;
          + (shape[k][1] - mapped[j][1]) * (shape[k][1] - mapped[j][1])/262144;
     if (dist < max1) {min = max1; break;}
     if (dist < min) min = dist;}
   max1 = min;}

for (k = 0, max2 = 0; k < points; k++)
  {for (j = 0, min = 262144; j < pixels; j+= SUB)
    {dist = (shape[j][0] - mapped[k][0]) * (shape[j][0] - mapped[k][0])/262144;
          + (shape[j][1] - mapped[k][1]) * (shape[j][1] - mapped[k][1])/262144;
     if (dist < max2) {min = max2; break;}
     if (dist < min) min = dist;}
   max2 = min;}

colour++;
if (max2 > max1) max1 = max2;
oldfit[n] = (int) (oldfit[n] * (256.0 - smax)/(256.0 * pixels * (1.0 + sqrt((float) max1))));
if (oldfit[n] > best) {best = oldfit[n]; write_best(best, map);}}
```

WRITE_BEST

Outputs the best-so-far solution in two different formats. IMAGE is an ascii file and gives the code and its fitness in a readable form. IMAGE.IFS contains just the raw IFS data and could be used directly by a pattern recognition program as one of its library codes.

value . . . . . fitness value of best solution.

data[ ] . . . . . - mapping coefficients.

```
void write_best(value, data)
int value; int (*data)[6];
{int i, j;
FILE *out1, *out2;
short number = MAP;

out1 = fopen("image", "w");
out2 = fopen("image.ifs", "w");
fprintf(out1, "%s", "image \n");
fwrite(&number, 2, 1, out2);

for (i = 0; i < MAP ;i++)
  for (j = 0; j < 4; j++) fprintf(out1, "%f", (float) (data[i][j])/256.0);
  for (j = 4; j < 6; j++) fprintf(out1, "%f", (float) (data[i][j])/262144.0);
  fprintf(out1, "%s", "\n");
  for (j = 0; j < 6; j++) fwrite(&data[i][j], 4, 1, out2);}
fprintf(out1, "%d\n", value);
fclose(out2); fclose(out1);}
```

RAND

Generates pseudo-random integers in the range $[a, b]$.

seed . . . . . seed of number generator.

```
int rand(a, b)
int a, b;
{float c;
static short seed;

{seed = ((int) seed * 25173 + 13849)%65536;
c = (float) seed/32768.0;
c = 0.5 * (b * c - a * c + c + b + a + 1.0);
return ((int) c);}
```

APPENDIX B

The following IFS codes are those used to test the GA implementation in chapter six. Using the normalised fitness functions as described in the text, the fitness of each of these exact codes is dependent on its minimum contractivity factor, s, and so this quantity is given for each case. We also supply the probabilities, $p(i)$ for $i = 1,2,\ldots,N$, used in the rendering of the codes, together with the number of points, n, that were produced.

Renderings of the attractors and representations of the collages for the set of test IFS's are given in the figures that follow directly after the tables of codes.

	a	b	c	d	e	f	$p(i)$
w_1 :	0.50	0.00	0.00	0.50	0.00	18.00	0.333
w_2 :	0.50	0.00	0.00	0.50	-15.00	-8.00	0.333
w_3 :	0.50	0.00	0.00	0.50	15.00	-8.00	0.333

Table B.1 *The IFS code for a Sierpinski triangle with* $s = 0.500$ *and* $n = 779$.

	a	b	c	d	e	f	$p(i)$
w_1 :	0.50	0.00	0.00	0.50	15.00	15.00	0.250
w_2 :	0.50	0.00	0.00	0.50	15.00	-15.00	0.250
w_3 :	0.50	0.00	0.00	0.50	-15.00	15.00	0.250
w_4 :	0.50	0.00	0.00	0.50	-15.00	-15.00	0.250

Table B.2 *The IFS code for a square with* $s = 0.500$ *and* $n = 3291$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.20	-0.26	0.23	0.22	0.00	24.00	0.111
w_2	:	-0.15	0.28	0.26	0.24	0.00	6.60	0.116
w_3	:	0.85	0.04	-0.04	0.85	0.00	24.00	0.773

Table B.3 *The IFS code for a fern with* $s = 0.851$ *and* $n = 4030$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.59	-0.37	0.37	0.59	60.00	0.00	0.500
w_2	:	0.59	-0.37	0.37	0.59	-60.00	0.00	0.500

Table B.4 *The IFS code for the twin-dragon fractal with* $s = 0.696$ *and* $p = 1809$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.00	0.50	0.50	0.00	15.00	15.00	0.333
w_2	:	0.50	0.00	0.00	0.50	15.00	-15.00	0.333
w_3	:	-0.50	0.00	0.00	0.50	-15.00	-15.00	0.333

Table B.5 *The IFS code for the 'L' shaped fractal with* $s = 0.500$ *and* $p = 1128$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.32	-0.16	0.25	0.49	5.60	19.96	0.236
w_2	:	0.53	-0.25	0.13	0.68	-12.43	16.98	0.472
w_3	:	-0.18	-0.83	0.33	0.18	14.25	10.14	0.292

Table B.6 *The IFS code for random fractal, rand1, with* $s = 0.849$ *and* $n = 880$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.01	-0.07	0.18	0.27	30.00	34.00	0.038
w_2	:	0.40	0.43	-0.53	0.22	24.00	-21.00	0.768
w_3	:	-0.03	0.51	-0.15	-0.12	-15.00	20.00	0.194

Table B.7 *The IFS code for random fractal, rand2, with* $s = 0.664$ *and* $n = 346$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.34	-0.21	0.21	0.34	-8.92	13.43	0.274
w_2	:	0.36	0.37	0.37	-0.36	20.41	6.29	0.456
w_3	:	0.39	0.07	-0.07	0.39	-23.61	-2.52	0.270

Table B.8 *The IFS code for random fractal, rand3, with* $s = 0.500$ *and* $n = 508$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.65	0.33	-0.12	0.56	13.45	-34.21	0.523
w_2	:	0.33	-0.50	0.66	0.11	-15.89	6.53	0.477

Table B.9 *The IFS code for random fractal, rand4, with* $s = 0.738$ *and* $n = 1316$.

		a	b	c	d	e	f	$p(i)$
w_1	:	0.25	0.33	-0.72	0.56	1.45	-14.21	0.538
w_2	:	0.33	-0.35	0.16	0.81	-15.89	16.53	0.462

Table B.10 *The IFS code for random fractal, rand5, with* $s = 0.882$ *and* $n = 1102$.

		a	b	c	d	e	f	$p(i)$
w_1	:	-0.34	0.21	-0.42	-0.40	5.63	-14.31	0.375
w_2	:	0.23	-0.55	0.21	0.66	-23.77	29.40	0.448
w_3	:	-0.33	0.59	-0.09	-0.16	30.12	-6.51	0.177

Table B.11 *The IFS code for random fractal, rand6, with* $s = 0.859$ *and* $n = 1032$.

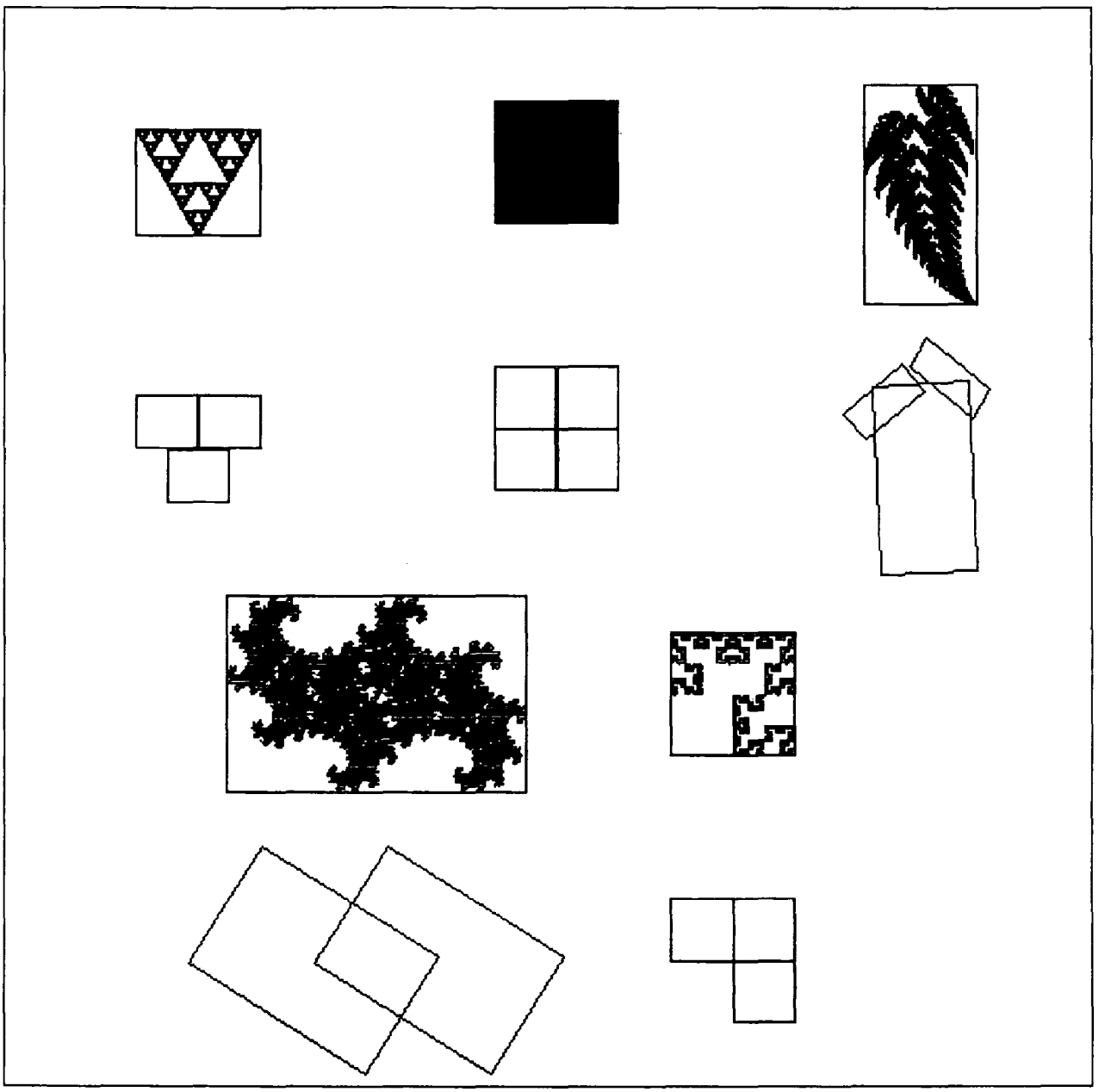


Figure B.1 *The collages and attractors for the first five IFS codes of the test set. (Top line from left) - The Sierpinski triangle, a square, and a fern. (Bottom left) - The twin-dragon fractal used by Barnsley. (Bottom right) - an 'L' shaped fractal.*

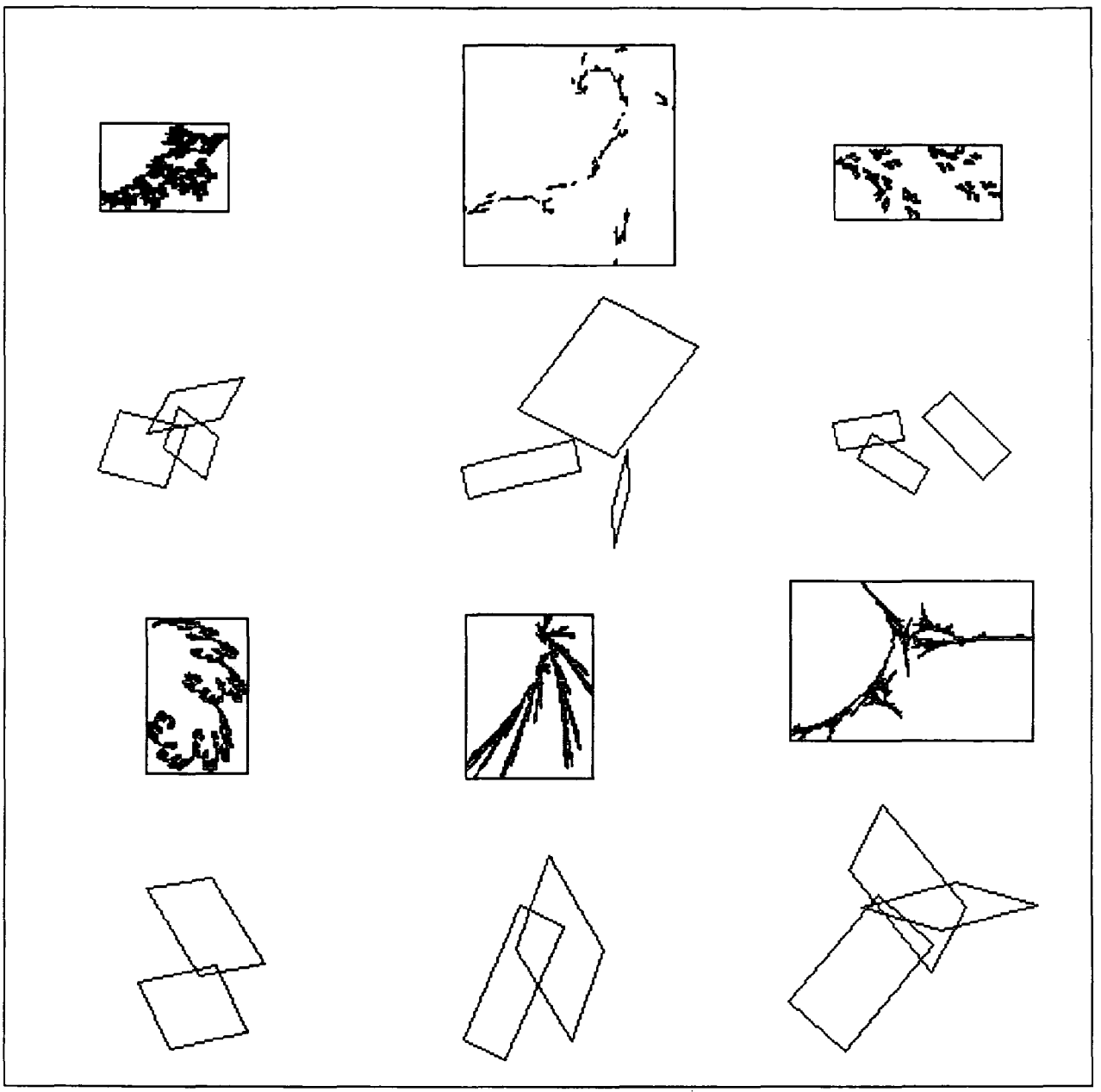
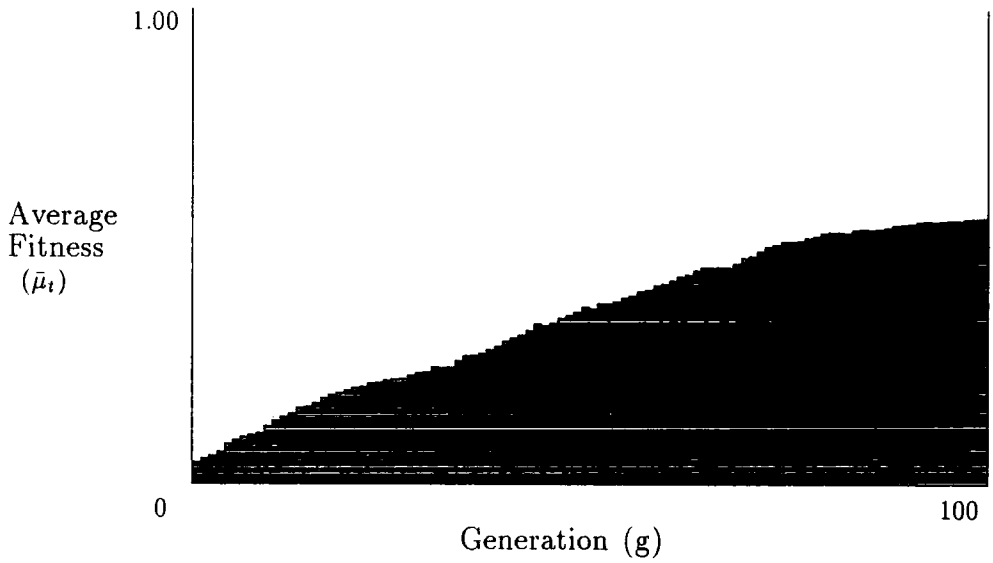
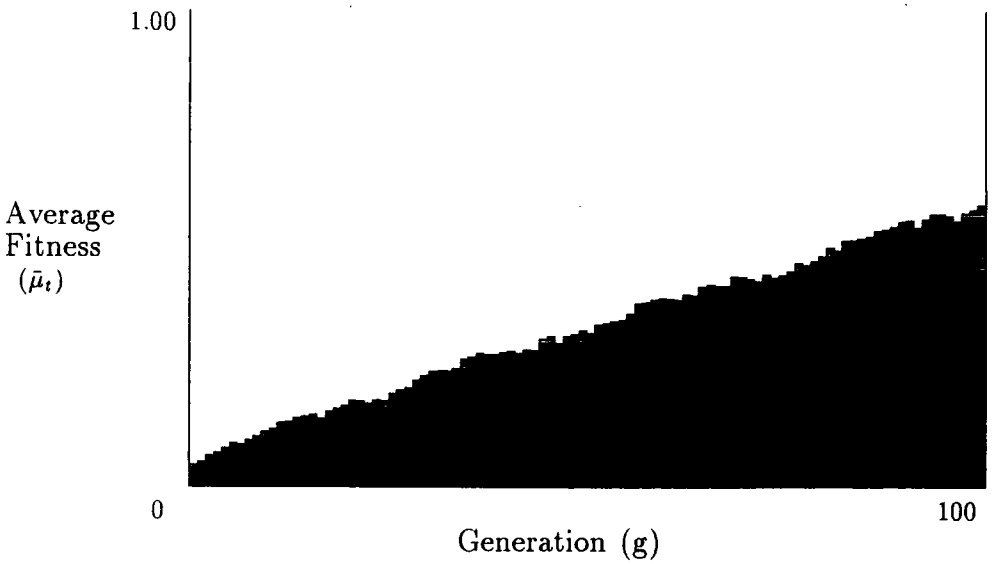


Figure B.2 *The collages and attractors for the randomly generated fractals in the test set. They are labelled rand1 to rand6 from top left to bottom right.*

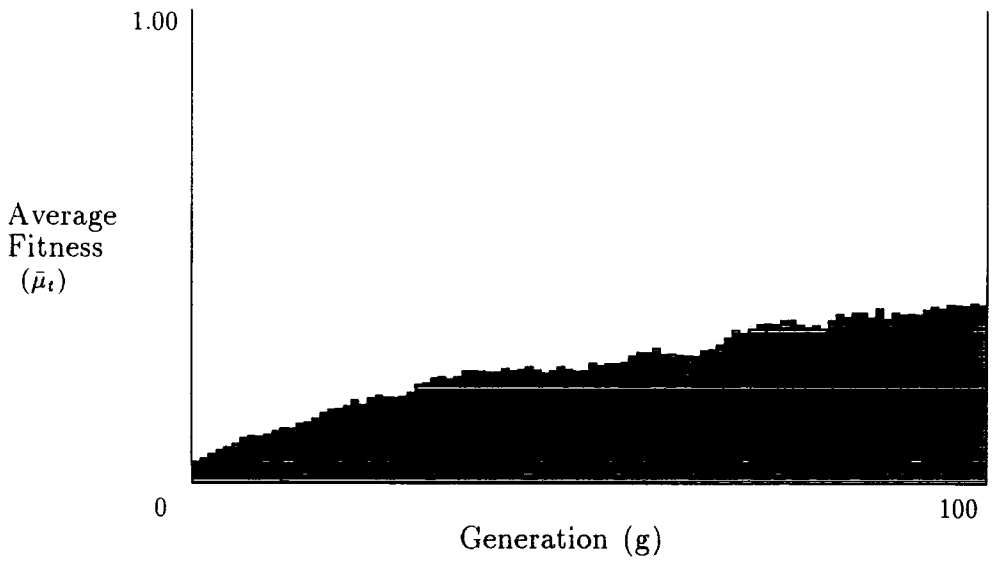
APPENDIX C



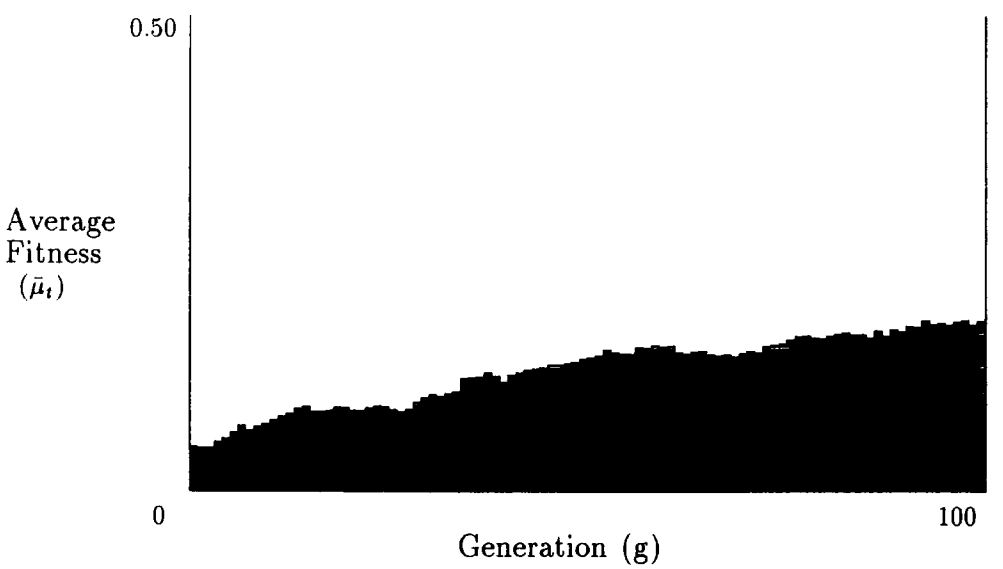
Graph C.1 *The increase in average population fitness over one hundred generations for zero mutation and other parameters fixed as described in the text.*



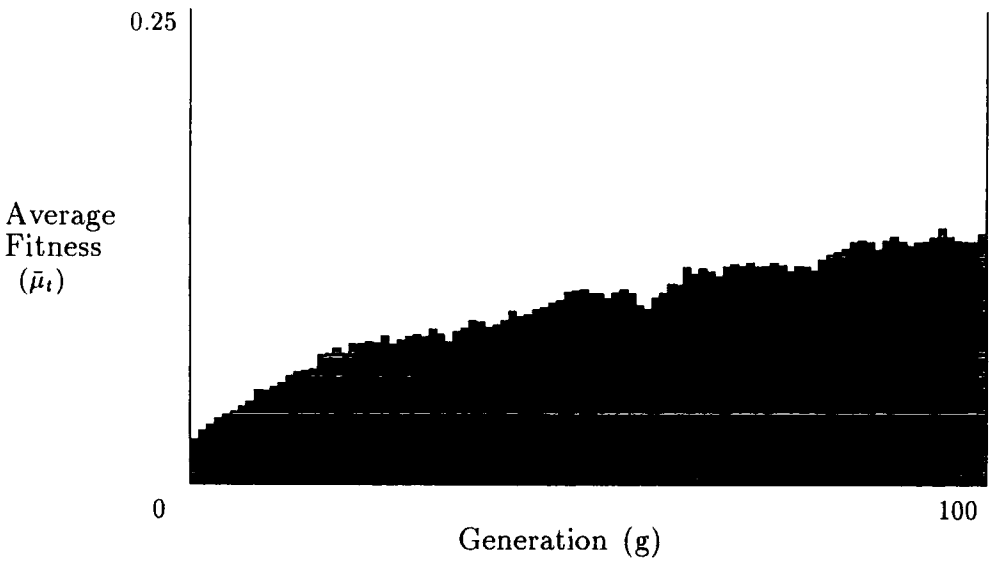
Graph C.2 *The effect on average population fitness of introducing a mutation probability of* 0.25 *whilst keeping all other parameters fixed.*



Graph C.3 *Increasing mutation rate to* $MUT = 0.5$ *shows severe impairment of performance.*

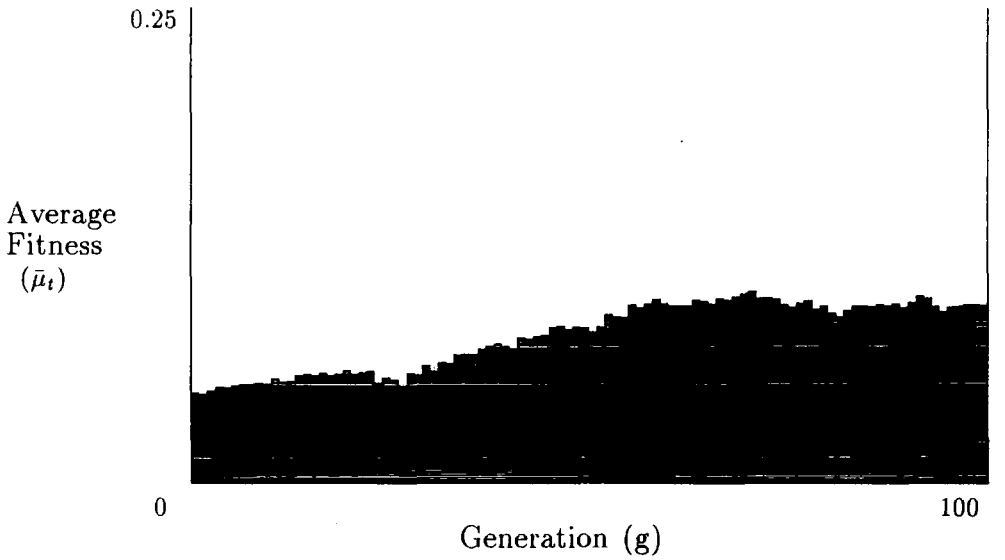


Graph C.4 *The effect of setting* $MUT = 0.75$. *(Notice the change of scale).*

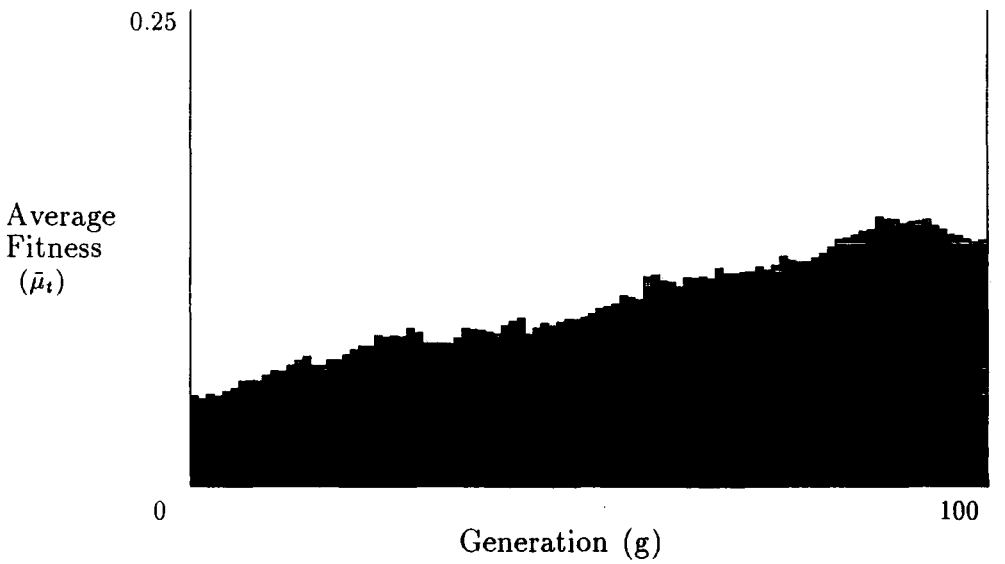


Graph C.5 *When all solutions are mutated the algorithm becomes little better than a random search.*

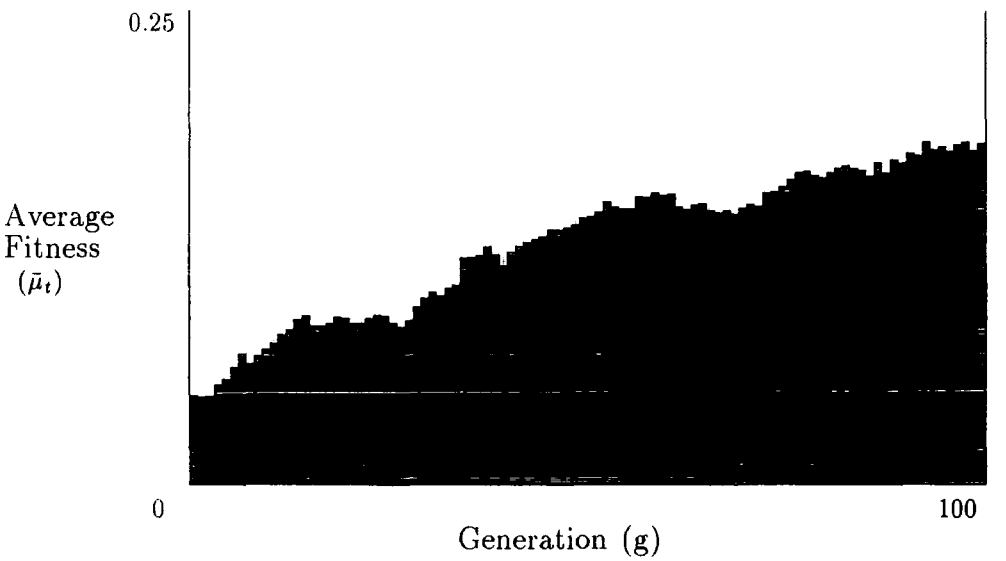
Graph C.6 *The finally selected value of* $MUT = 0.01$ *which maintains diversity without impairing the smooth increase of average fitness.*



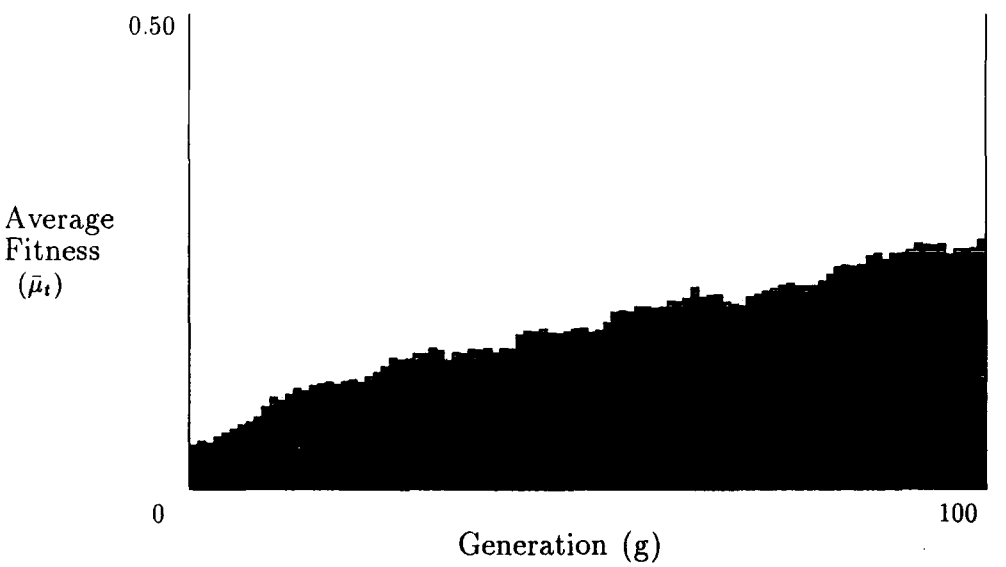
Graph C.7 *The result of setting $XLN = 0$ whilst retaining a mutation rate of one percent.*



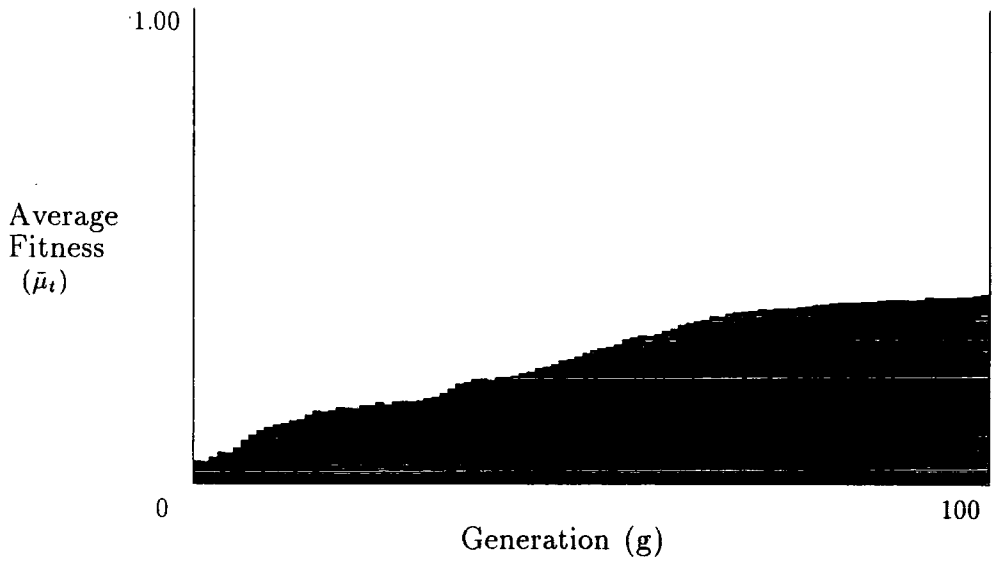
Graph C.8 *Inreasing the crossover length to two gives only slightly improved performance.*



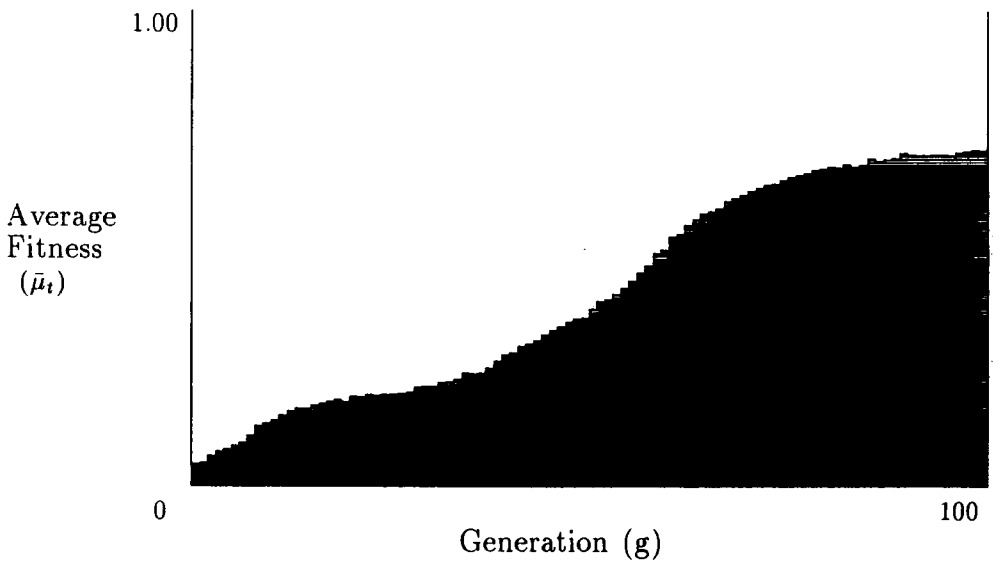
Graph C.9 *$XLN = 4$ and the underlying increase in average fitness starts to become apparent.*



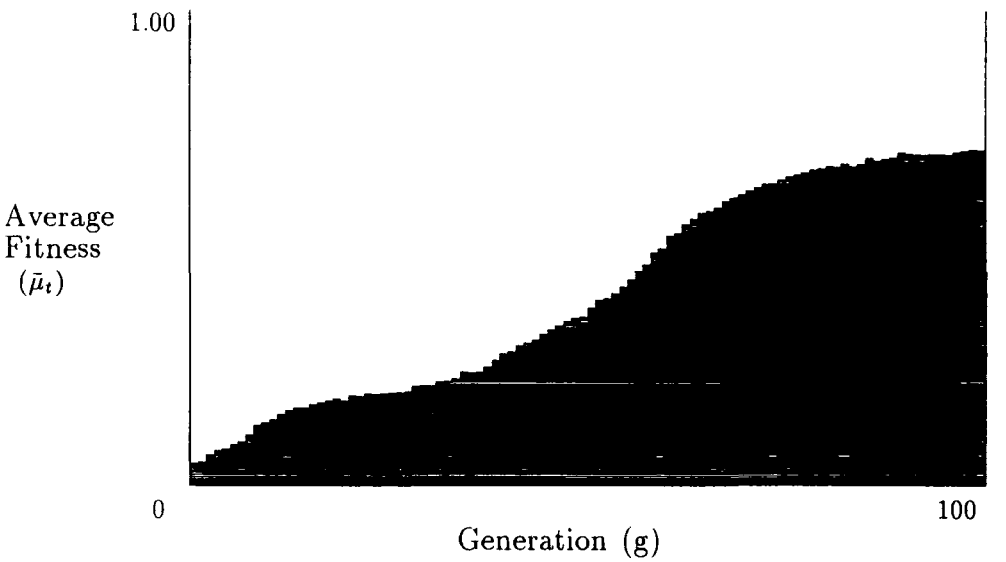
Graph C.10 *With the crossover length set at half of the full string length, the graph begins to smooth out and the absolute fitness values rise appreciably.*



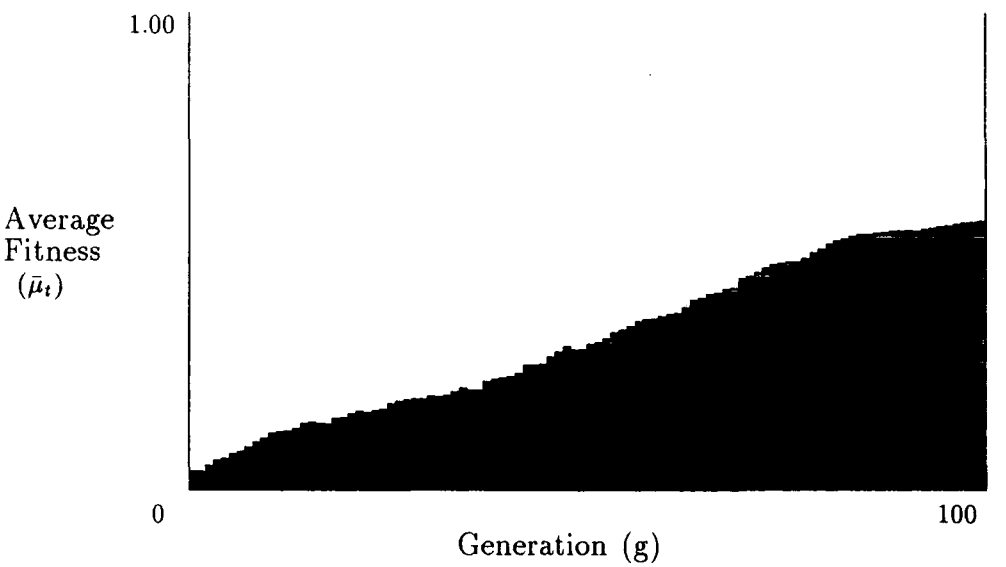
Graph C.11 *$XLN = 12$ and a smoothly increasing plot emerges.*



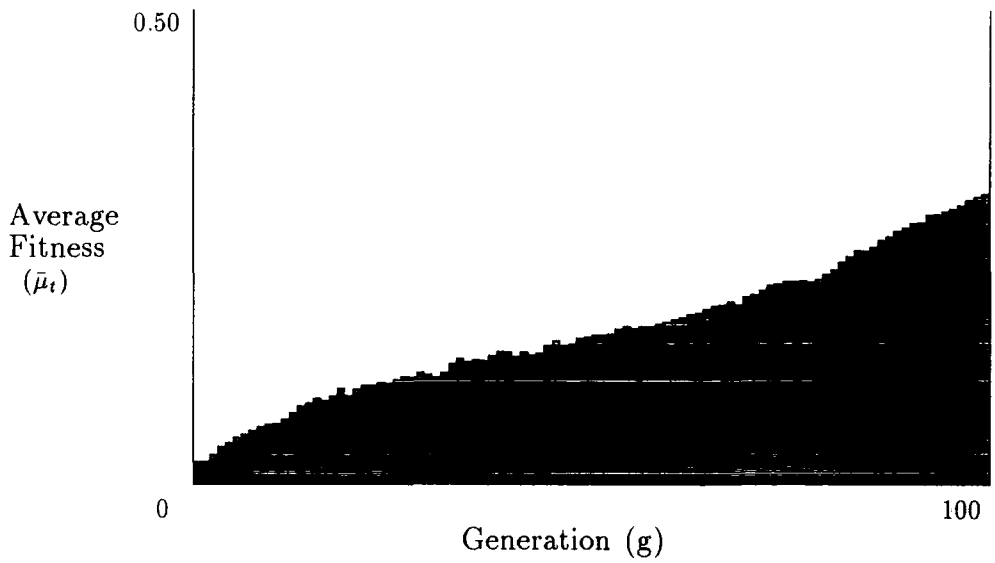
Graph C.12 *Allowing crossover to extend over the whole length of a solution clearly leads to the best performance within the imposed constraints.*



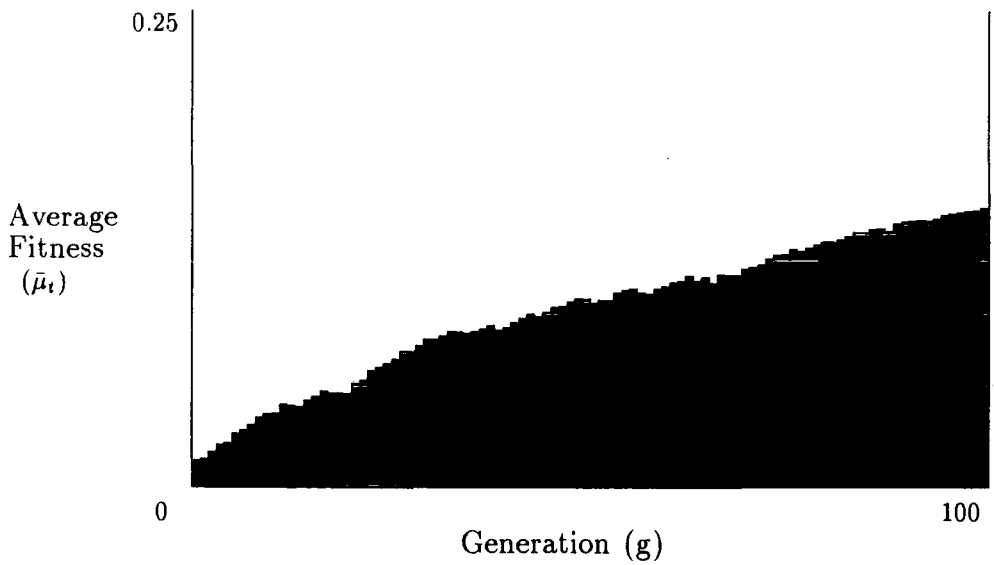
Graph C.13 *Average population fitness increase using accurate evaluation for each solution.*



Graph C.14 *The effect of subsampling the input shape by a factor of two.*

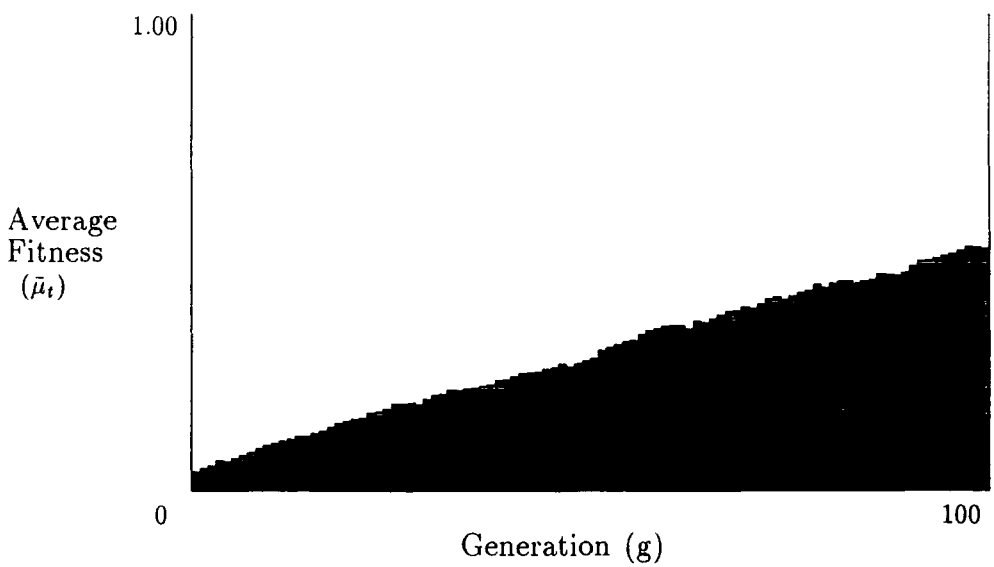


Graph C.15 *SUB = 5 and detrimental effects become apparent.*

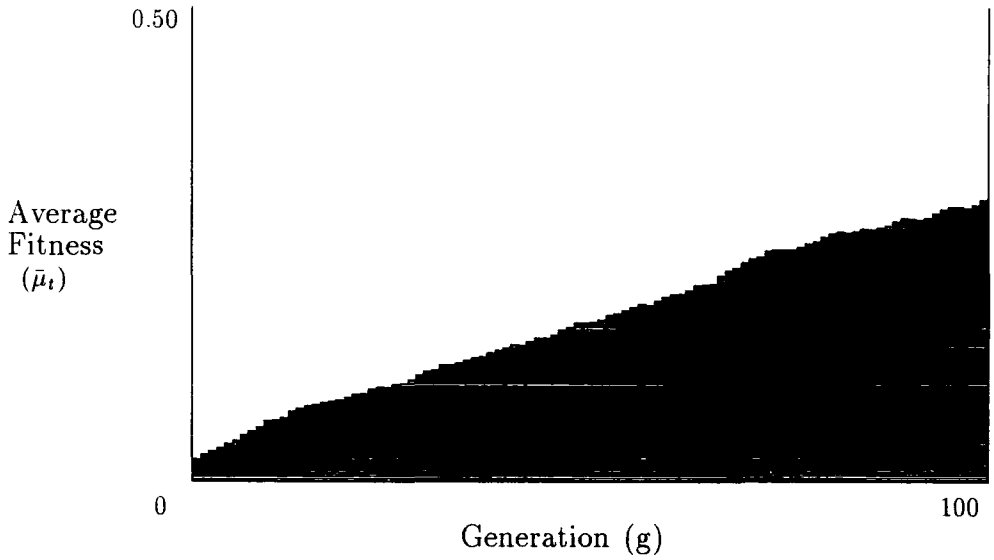


Graph C.16 *Taking only every tenth image point results in rapid convergence to a low average fitness population.*

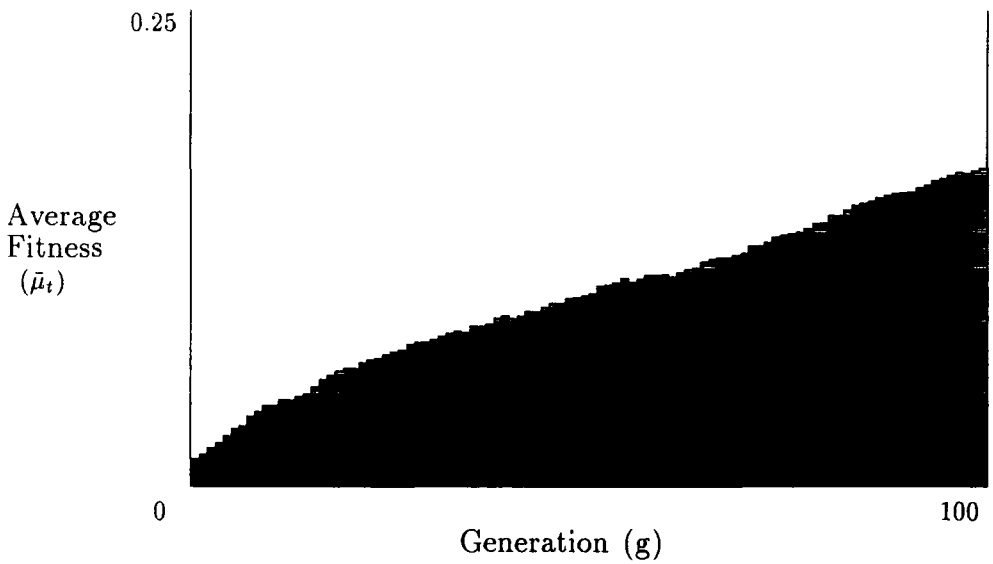
Graph C.17 *Fitness increase for accurate evaluation with a population size of one hundred.*



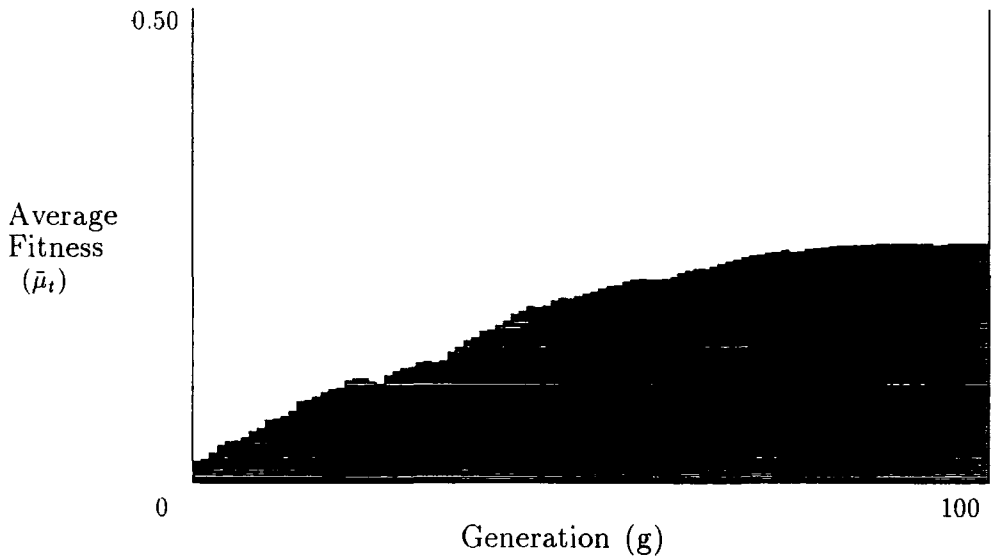
Graph C.18 *The change in performance obtained using a population of* 175 *with a subsampling factor of* 2.



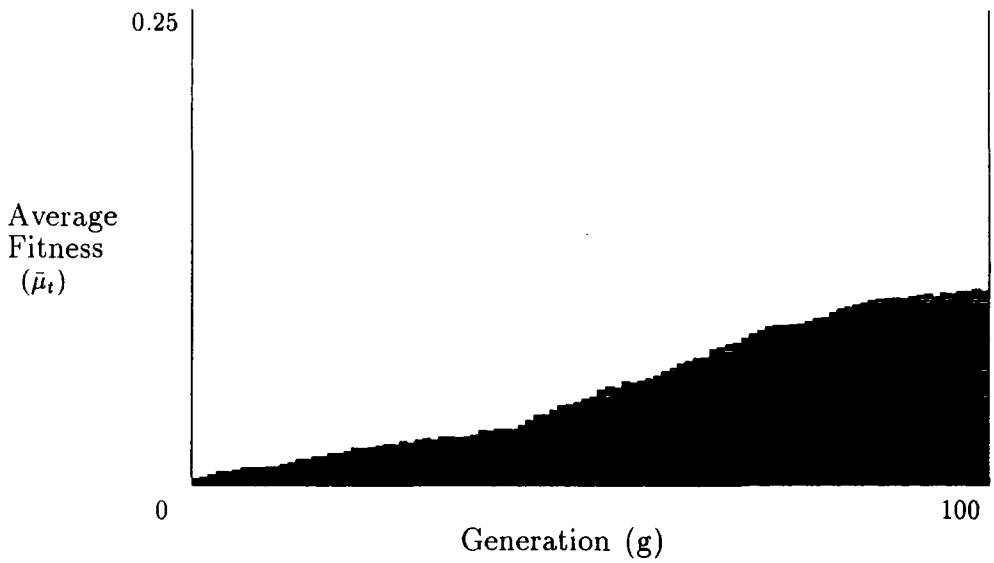
Graph C.19 *Results of using* $POP = 310$ *and* $SUB = 5$.



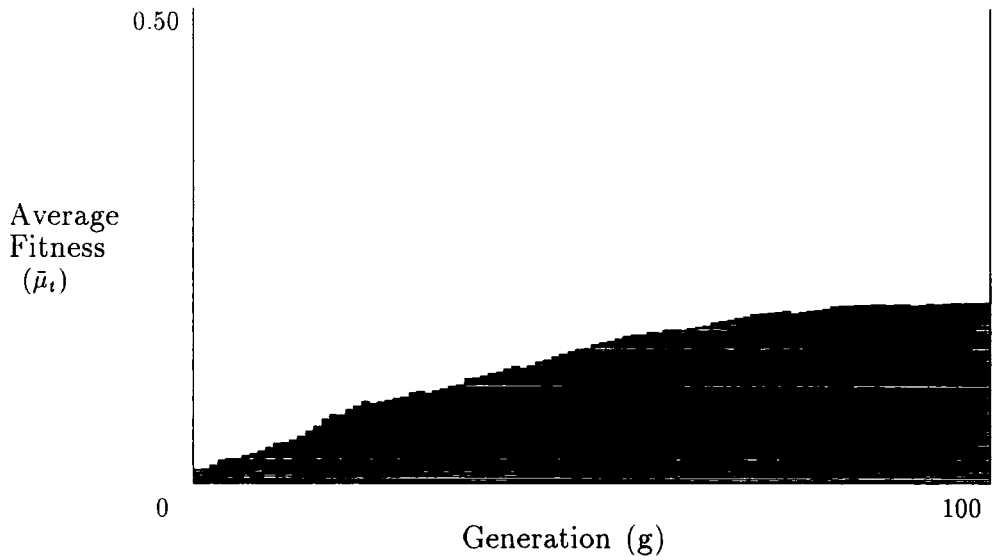
Graph C.20 *The performance obtained with a population of* 417 *and a subsampling factor of* 10.



Graph C.21 *Average population fitness as a function of generation for the first random fractal and fitness function B.*

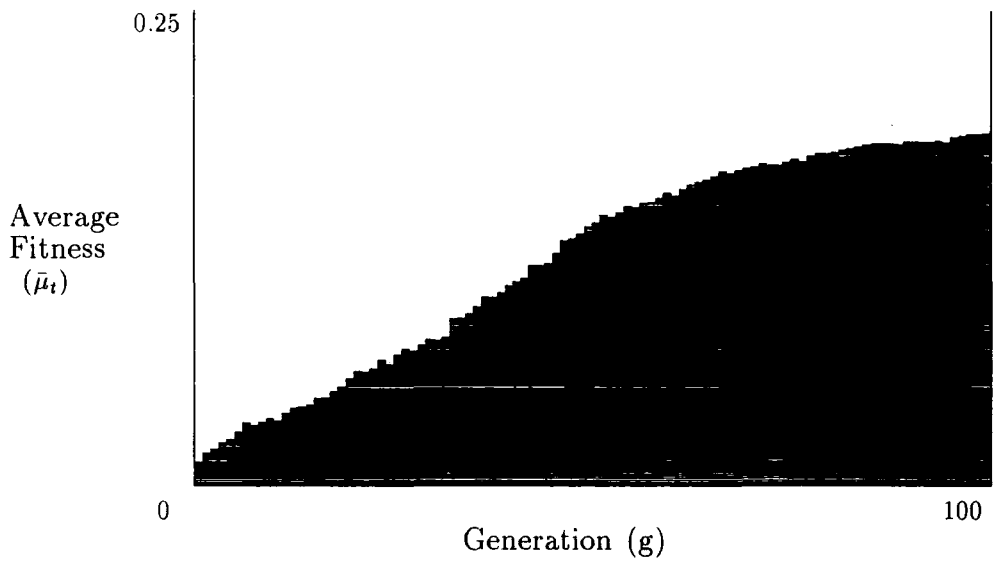


Graph C.22 *Average population fitness as a function of generation for the second random fractal and fitness function B.*

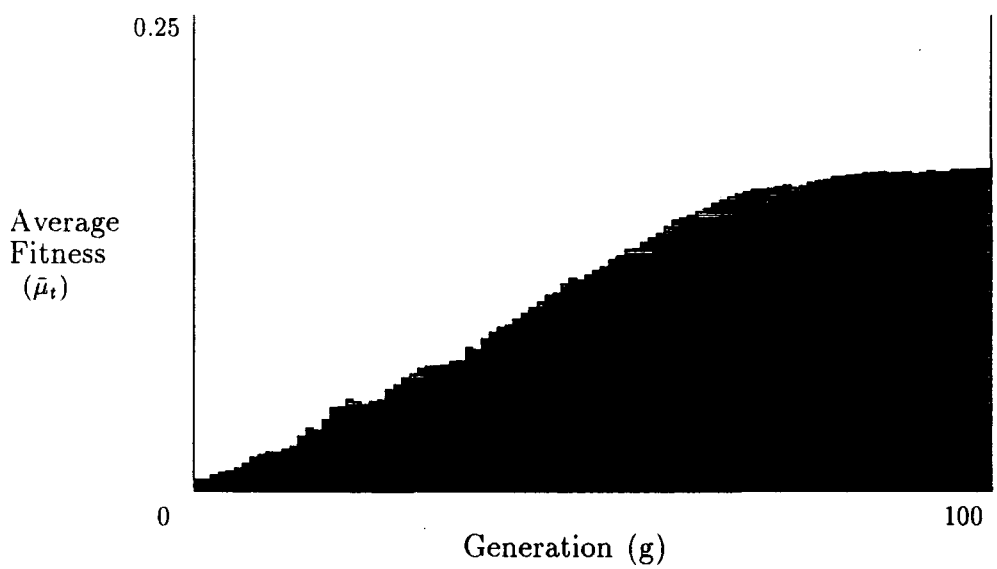


Graph C.23 *Average population fitness as a function of generation for the third random fractal and fitness function B.*

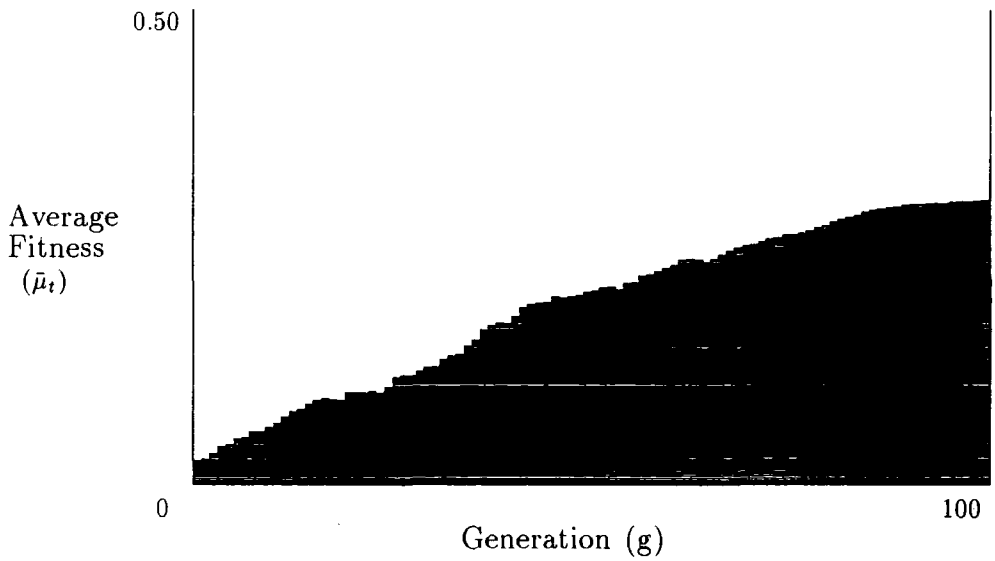
Graph C.24 *Average population fitness as a function of generation for the fourth random fractal and fitness function B.*



Graph C.25 *Average population fitness as a function of generation for the fifth random fractal and fitness function B.*

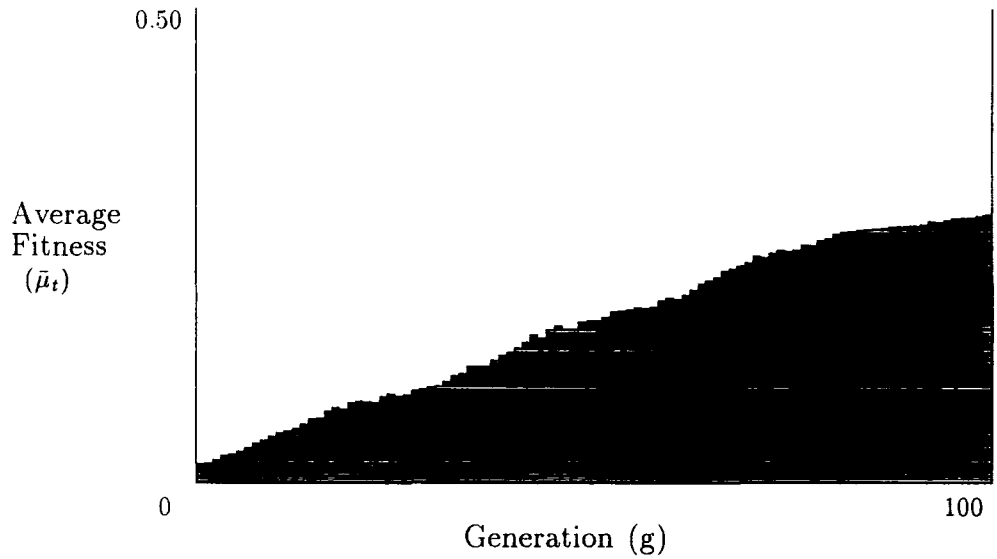


Graph C.26 *Average population fitness as a function of generation for the sixth random fractal and fitness function B.*

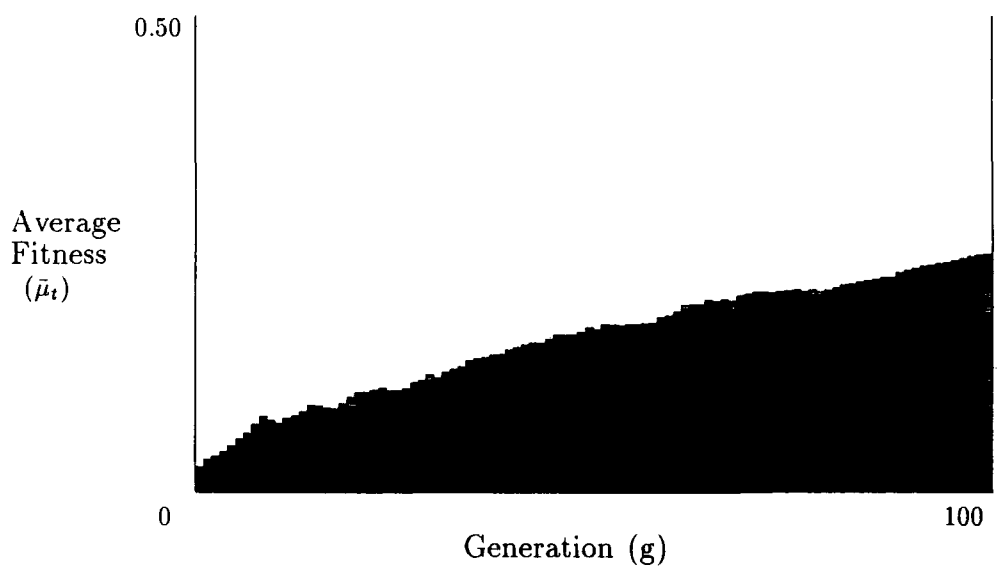


Graph C.27 *Average population fitness as a function of generation for the fern fractal and fitness function B.*

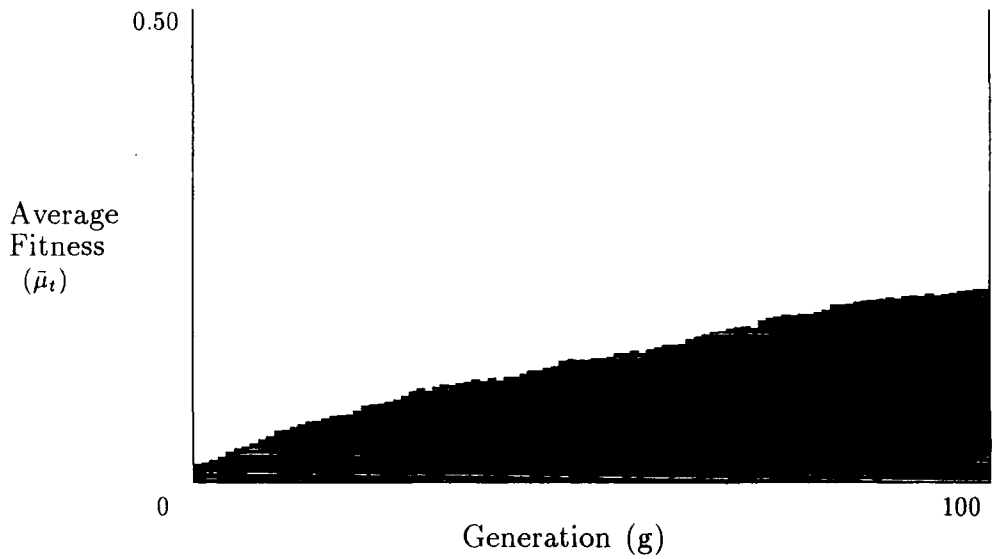
Graph C.28 *Average population fitness as a function of generation for the square test shape and fitness function B.*



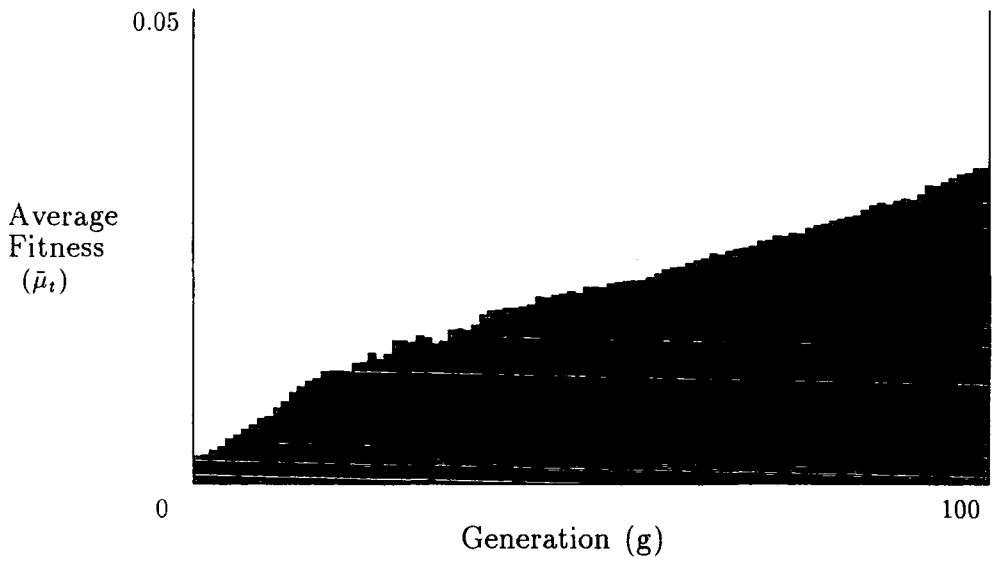
Graph C.29 *Average population fitness as a function of generation for the Sierpinski triangle and fitness function B.*



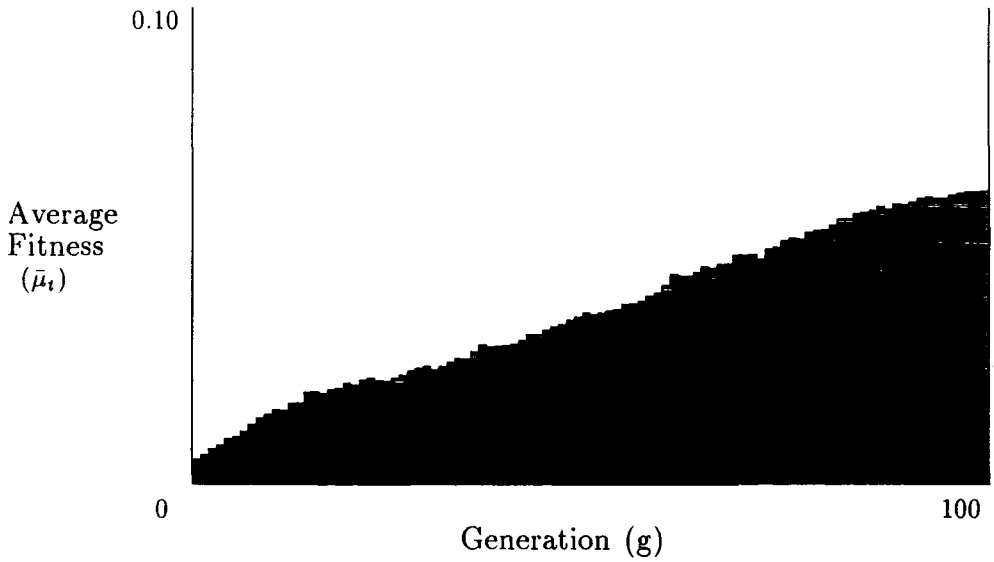
Graph C.30 *Average population fitness as a function of generation for the twin-dragon fractal and fitness function B.*



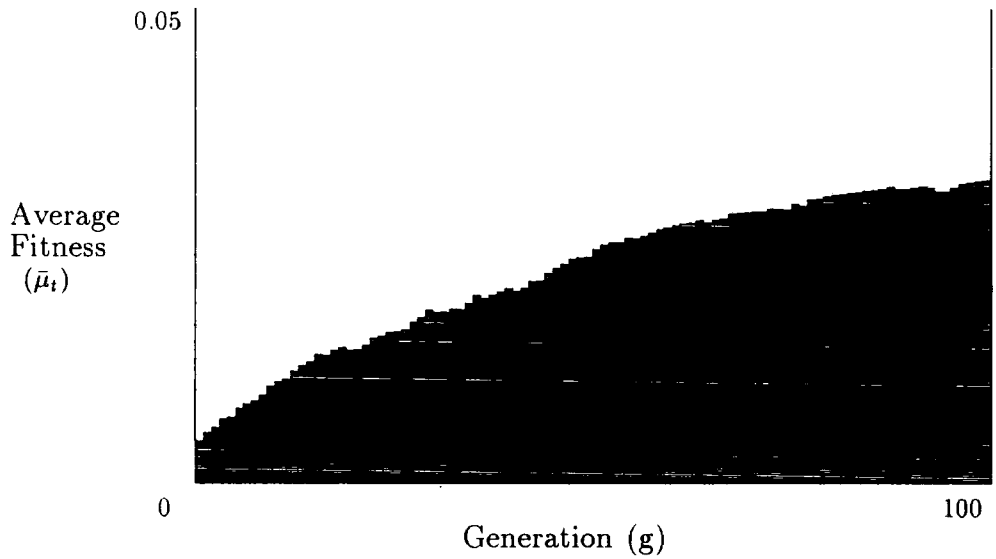
Graph C.31 *Average population fitness as a function of generation for the 'L' test shape and fitness function B.*



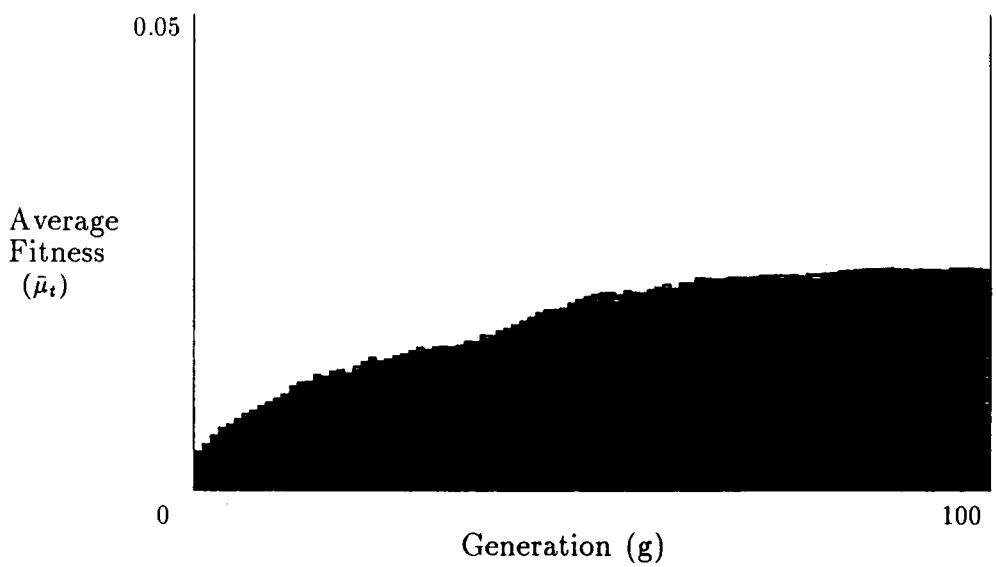
Graph C.32 *Average population fitness as a function of generation for the second random fractal and fitness function A.*



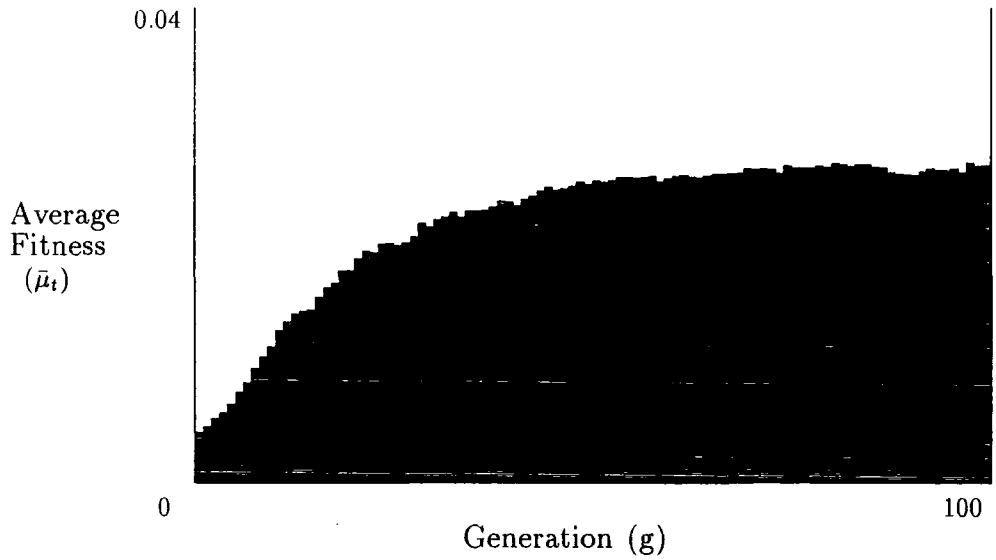
Graph C.33 *Average population fitness as a function of generation for the third random fractal and fitness function A.*



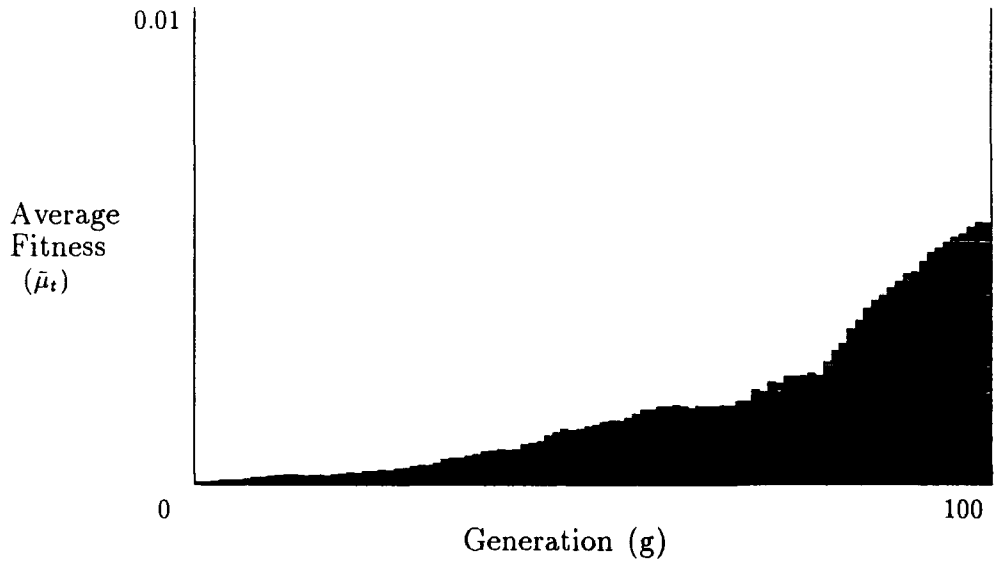
Graph C.34 *Average population fitness as a function of generation for the fourth random fractal and fitness function A.*



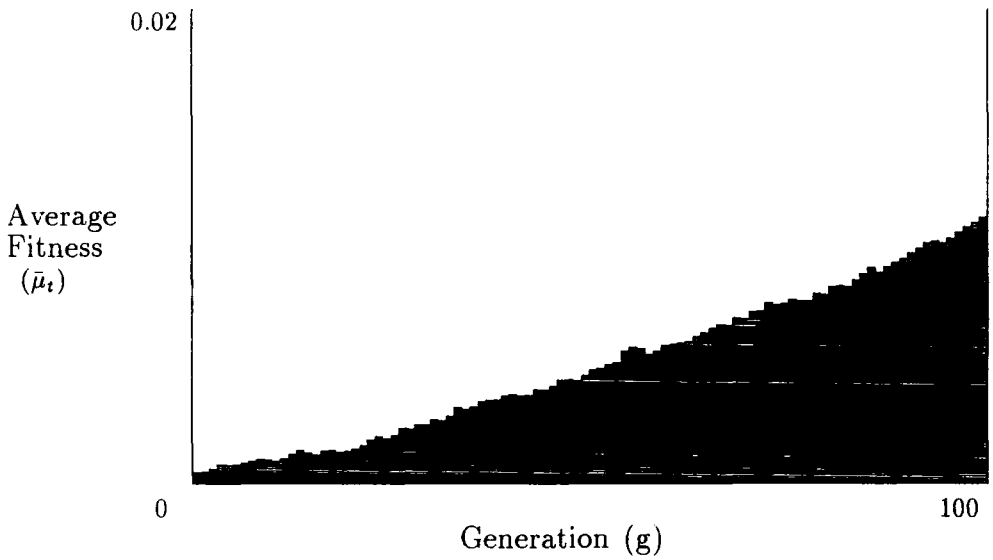
Graph C.35 *Average population fitness as a function of generation for the fifth random fractal and fitness function A.*



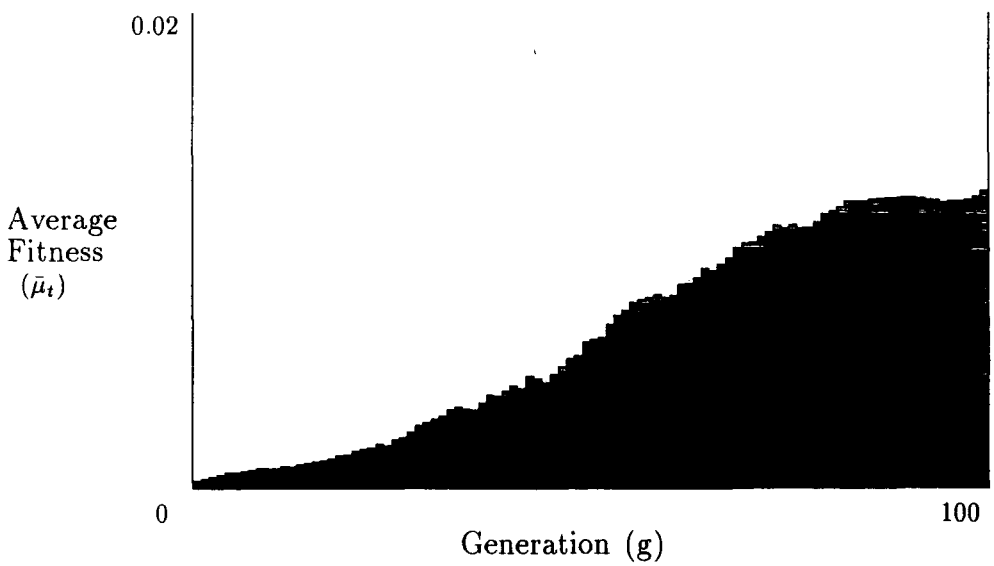
Graph C.36 *Average population fitness as a function of generation for the Sierpinski triangle and fitness function A.*



Graph C.37 *Average population fitness as a function of generation for the second random fractal and fitness function C.*

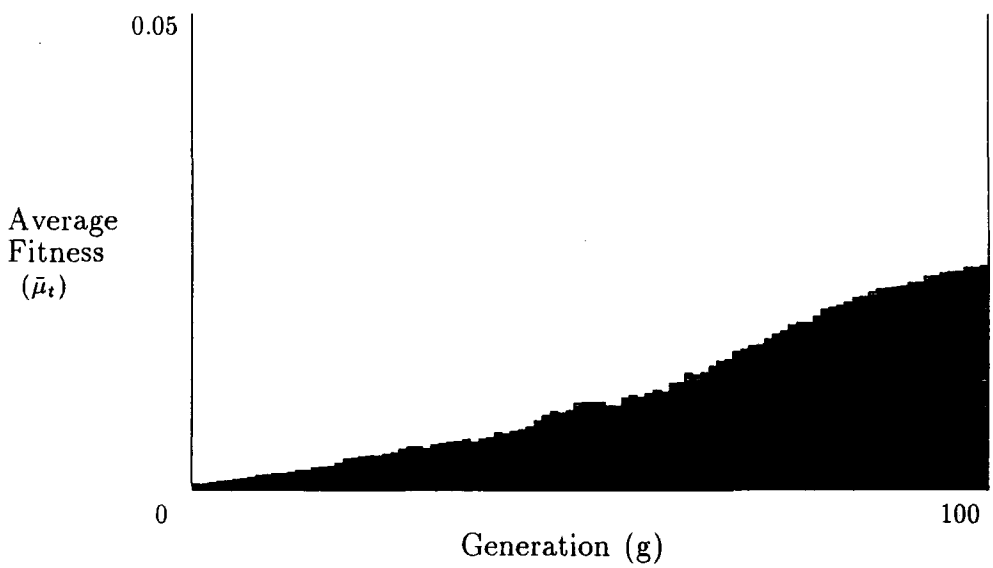


Graph C.38 *Average population fitness as a function of generation for the third random fractal and fitness function C.*

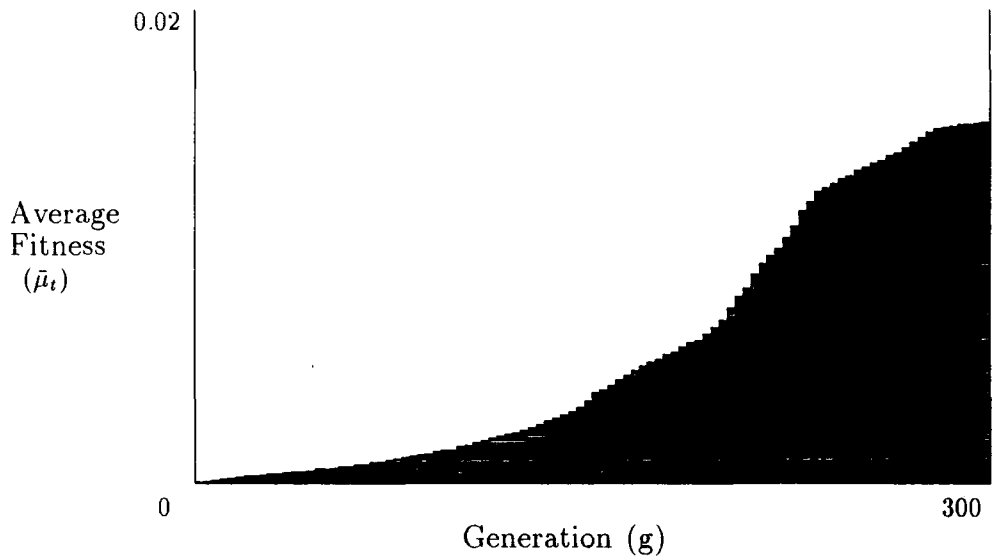


Graph C.39 *Average population fitness as a function of generation for the fourth random fractal and fitness function C.*

Graph C.40 *Average population fitness as a function of generation for the fifth random fractal and fitness function C.*



Graph C.41 *Average population fitness as a function of generation for the Sierpinski triangle and fitness function C.*



Graph C.42 *Average population fitness as a function of generation for the second random fractal, fitness function C, and a population size of* 1000.